PE
Practical Electronics

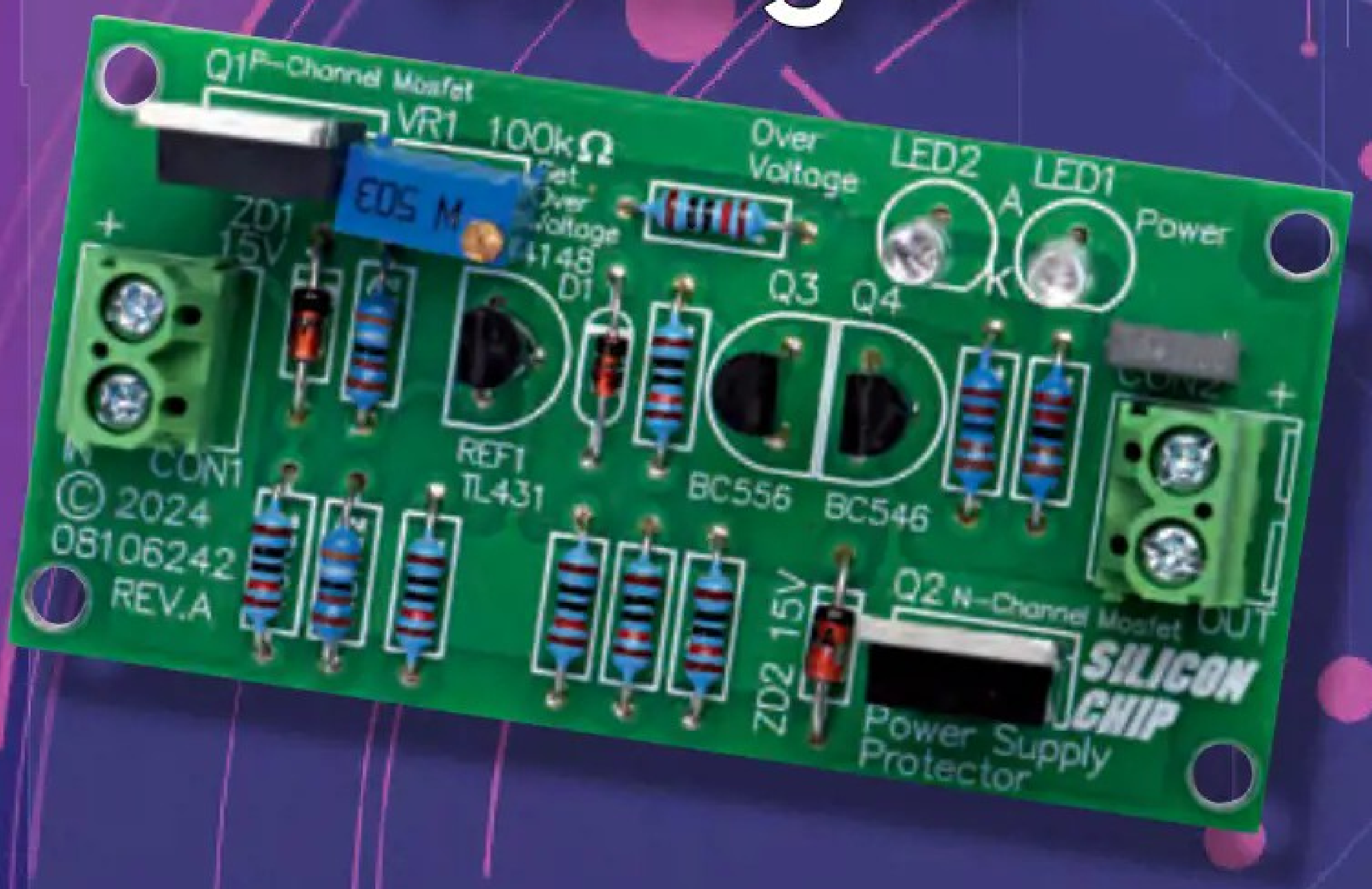
The UK's premier electronics, computing and maker magazine

Circuit Surgery
Digital high-pass, band-pass & band-stop filters

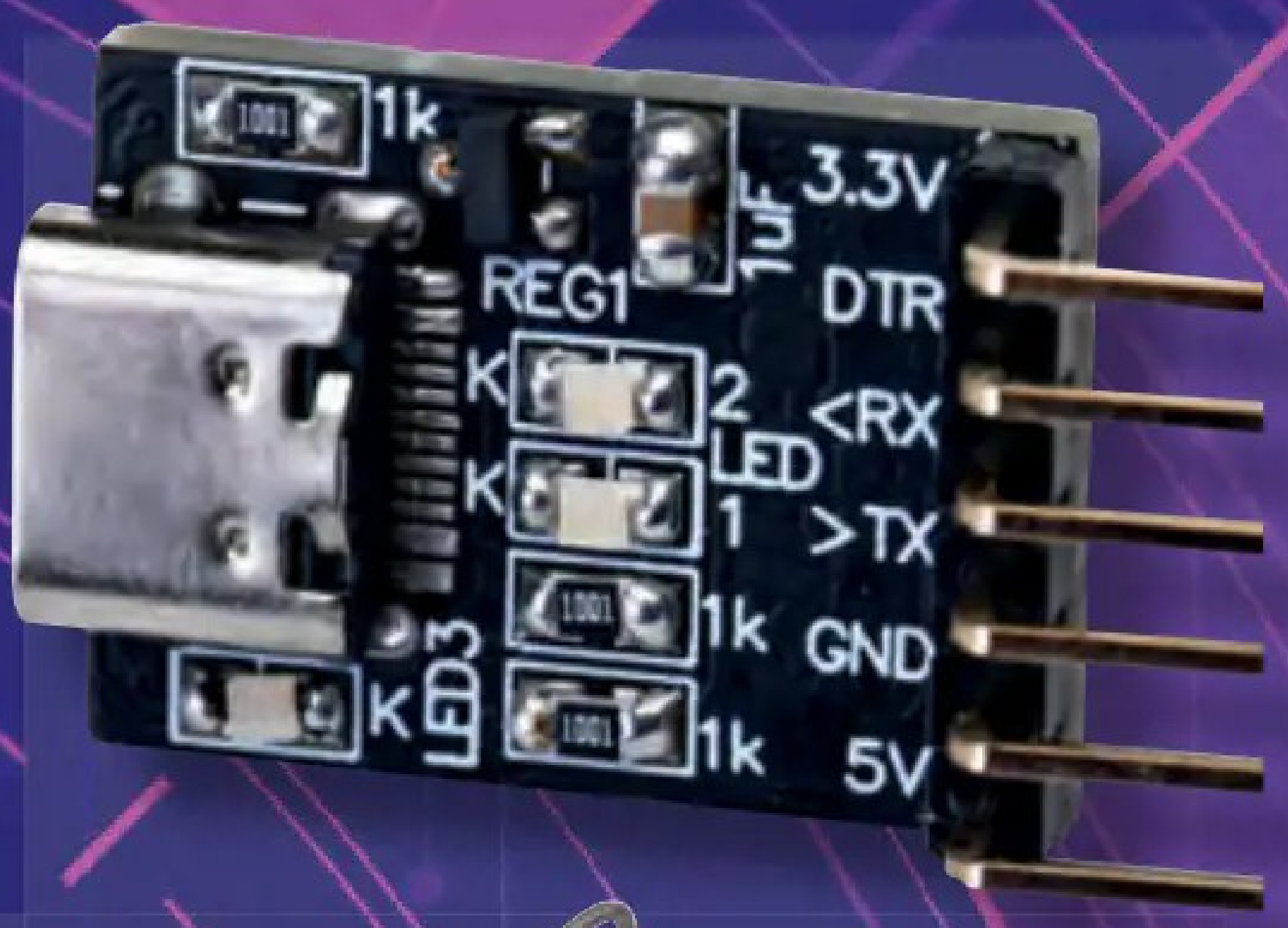
Audio Out
An amplifier that switches between Class-B and Class-A

Precision Electronics
Quantifying and dealing with noise

Protecting devices from damaging voltages


Q1 P-Channel Mosfet
VR1 100kΩ
Set Over Voltage
Over Voltage
LED2
LED1
Power
ZD1 15V
D1
Q3
Q4
CON1
IN
REF1 TL431
BC556
BC546
Q2 N-Channel Mosfet
OUT
ZD2 15V
© 2024
08106242
REV.A
SILICON CHIP
Power Supply Protector

DIY USB-C serial adaptor


1k
1uF
3.3V
REG1
DTR
K
2
LED
<RX
K
1
>TX
1k
GND
LED3
K
1k
5V

HEALTH
TIME
DURACELL

Skill Tester 9000
A new take on an old game

PLUS!
Techno Talk – the latest electronics technology
Cool Beans – creating & optimising display algorithms
Sinclair Calculators – 1970s innovation in the UK
Adding solar charging to a Renault Kangoo electric van

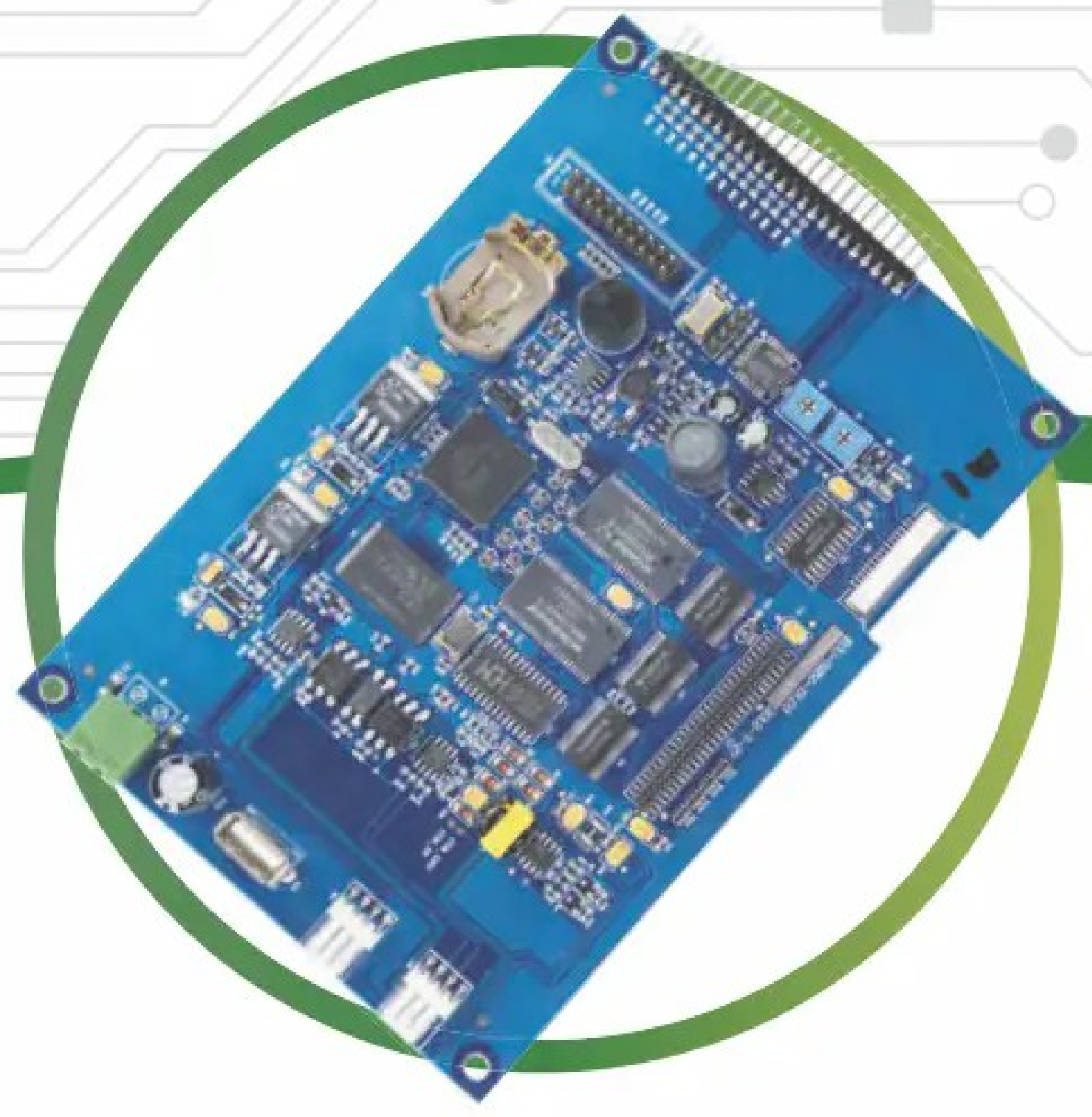
May 2025 £5.99

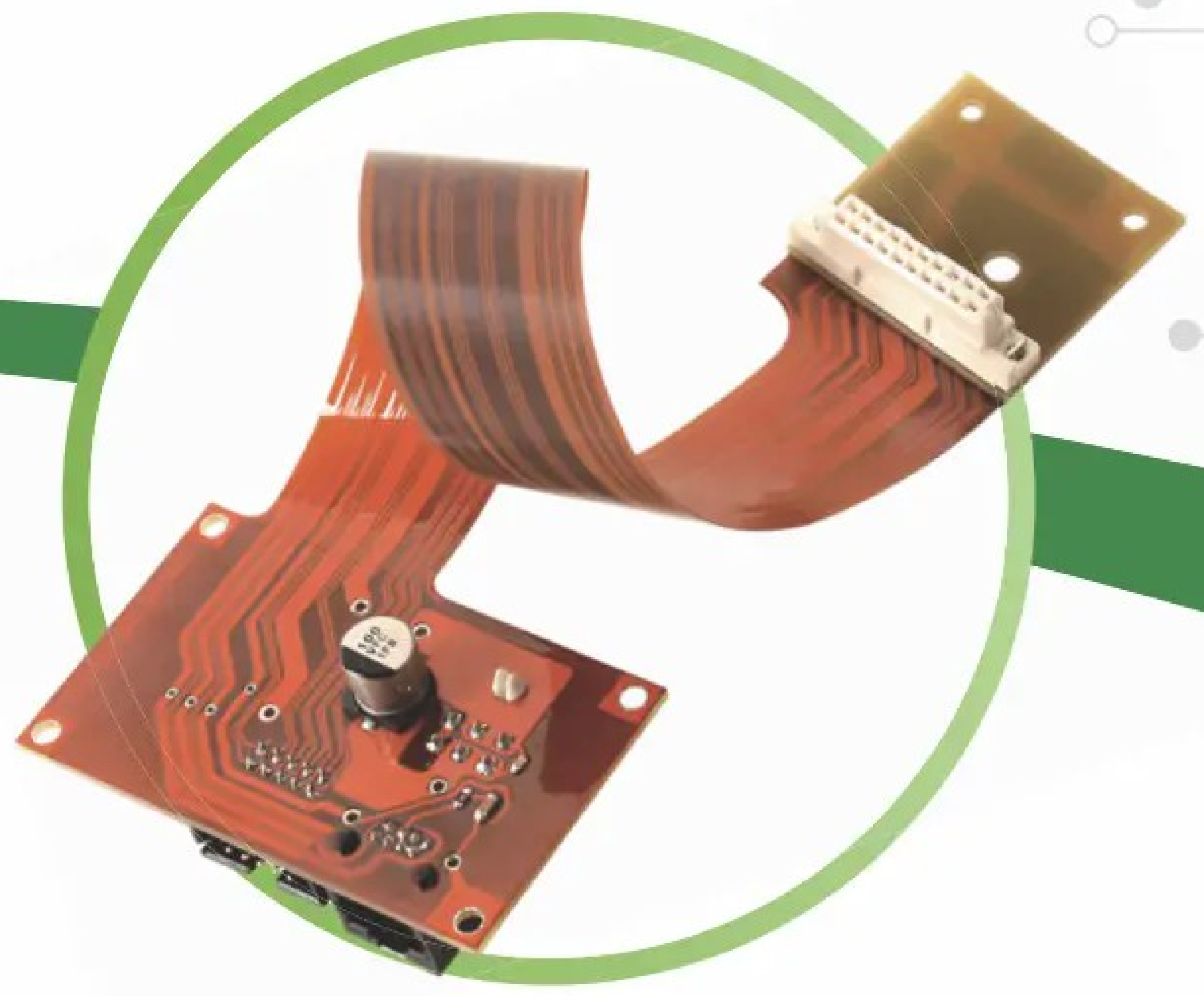
www.electronpublishing.com | @practicalelec | practicalelectronics

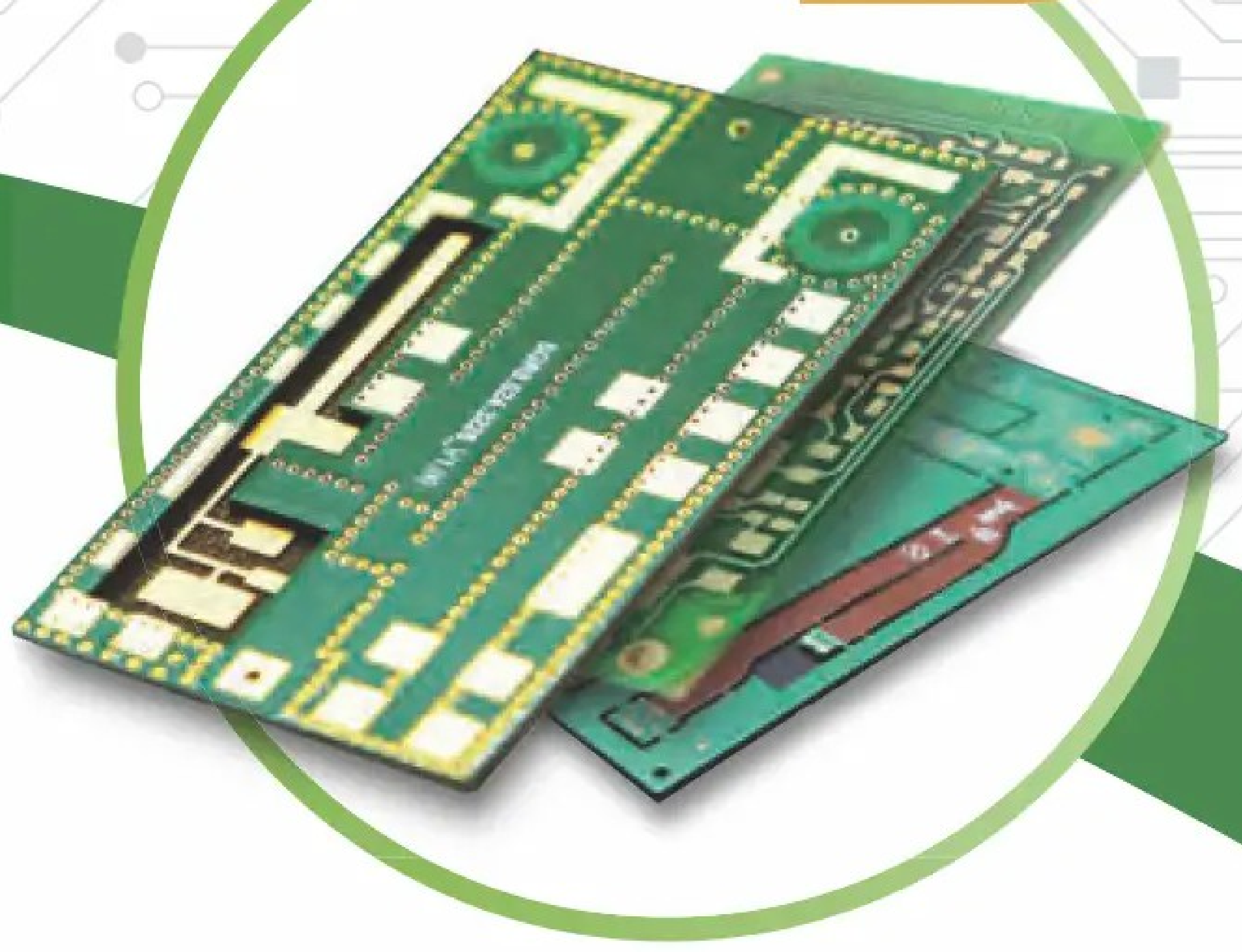
PCBWay

FULL-SERVICE
ELECTRONICS
MANUFACTURER

PCB Capabilities:







Special price!

PCB Assembly
From $29
- Improved assembly precision
- Short turnaround for assembly
- Flexible support

Flex, Rigid-Flex PCB
From $46.74
- Up to 16 layers Flex circuits
- Polymide Flex, PET, High Frequency
- 0.025-≥0.4 mm FPC thickness

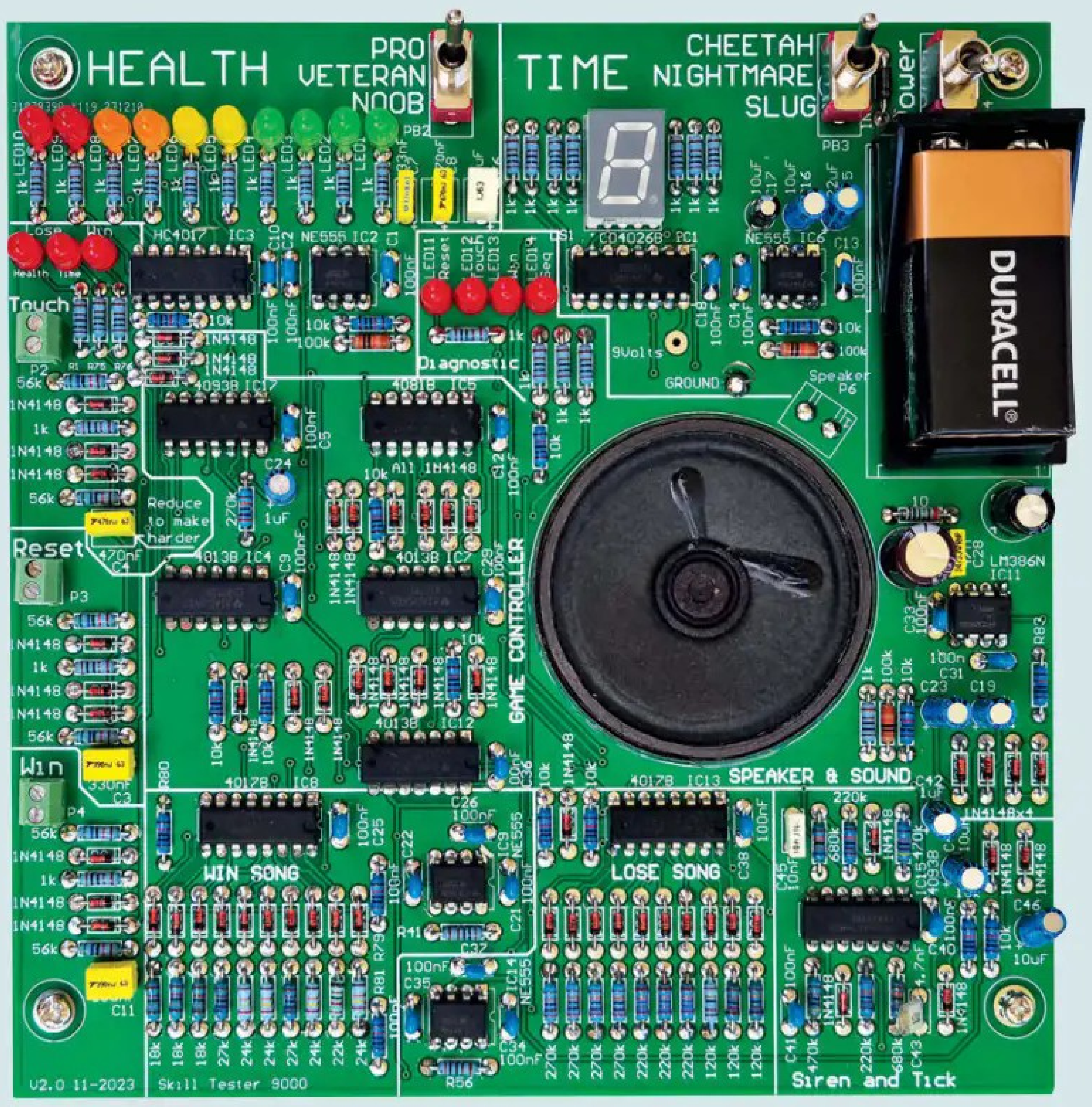
PCB Prototype
From $5/10 pcs
- Multiple color options
- 24-hour production time
- Worldwide delivery

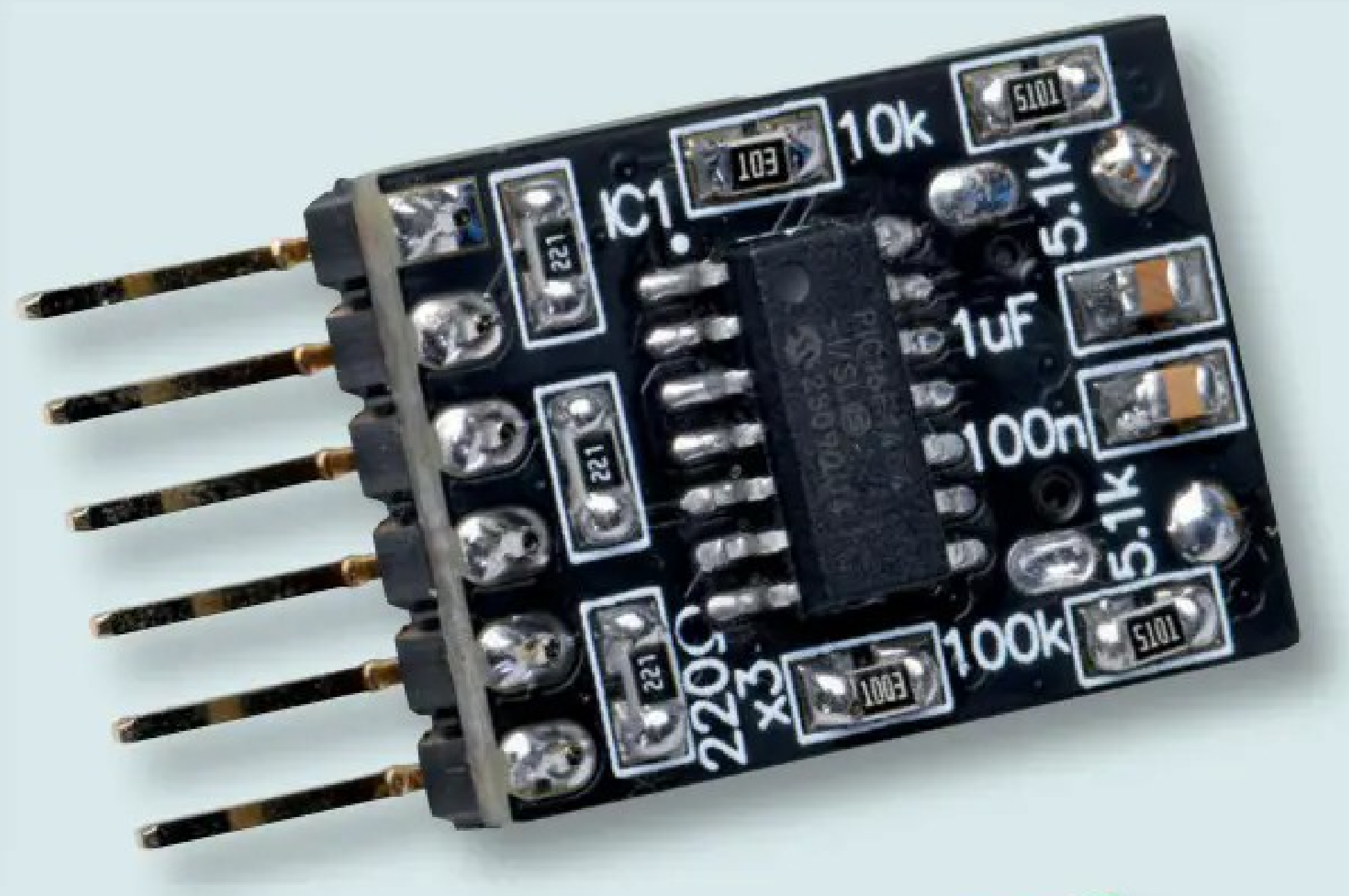
www.pcbway.com
service@pcbway.com

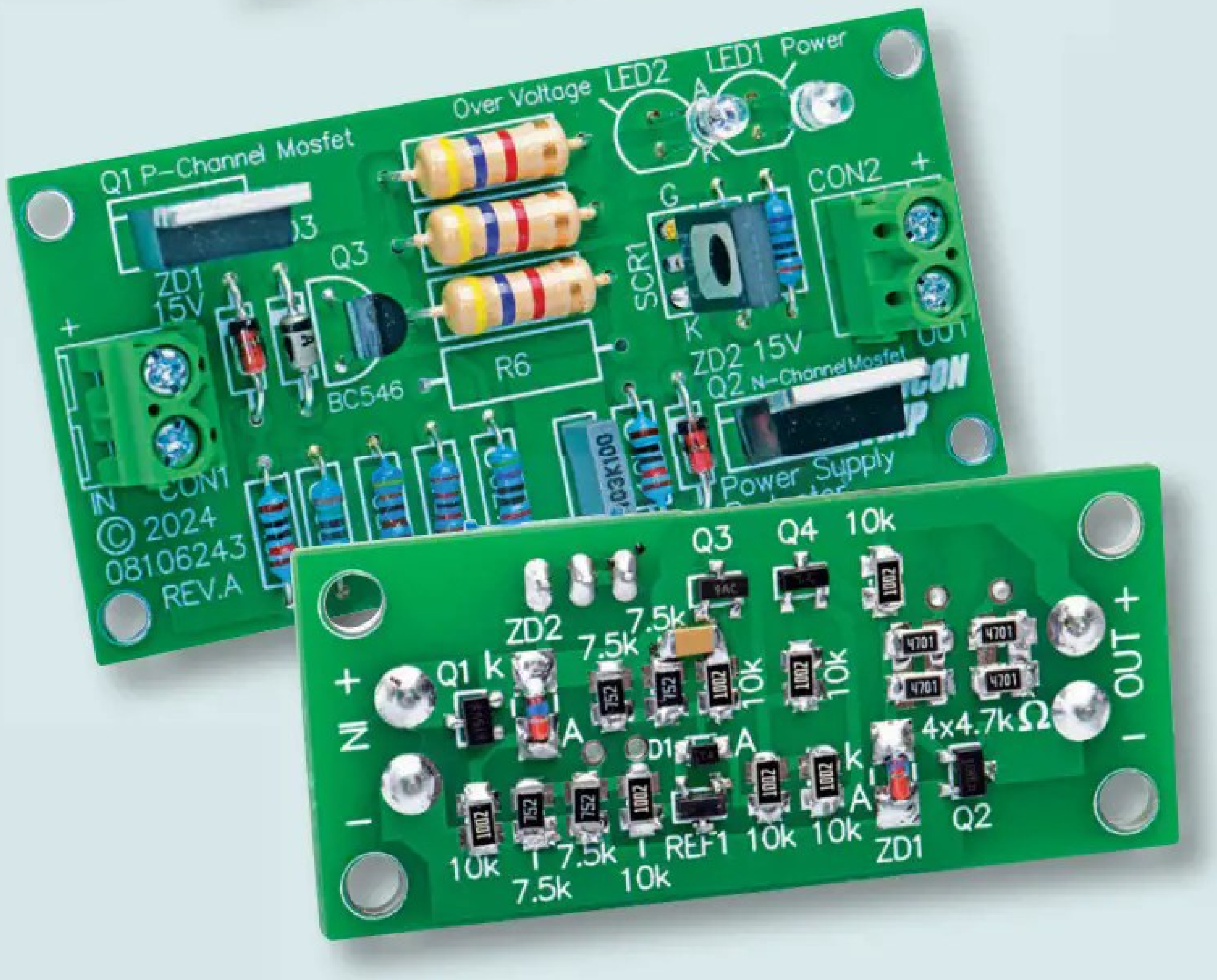
Practical
Electronics

Volume 54. No. 5
May 2025
ISSN 2632 573X

Contents







Projects and Circuits

The Skill Tester 9000, part one by Phil Prosser 8
This old-school dexterity tester has added lights, timers, countdowns, sounds, noises and competition between players! Plus, it has plenty of construction fun, and you can modify it to your heart's content

DIY USB-C Serial Adaptor by Tim Blythman 30
While USB-serial adaptors with Type-C sockets are available, many do not adhere to the USB-C standard and may also have Windows driver problems. Our design, presented here, has no such drawbacks and is relatively simple and compact.

Three DC Supply Input Protectors by John Clarke 46
These three simple, inexpensive circuits protect equipment from damage due to an incorrectly connected or malfunctioning power supply. They protect against a higher than expected voltage or a reverse polarity supply.

Series, Features and Columns

Techno Talk by Max The Magnificent 3
I chat with several companies about their latest and greatest technological offerings.

A short history of Sinclair's handheld calculators by Chris Morris 4
In the 1970s, UK company Sinclair Scientific were leaders in calculator innovation.

Circuit Surgery by Ian Bell 18
Topics in digital signal processing – high-pass, band-pass and band-stop filters.

Raspberry Pi 5 review by Tim Blythman 25
The Raspberry Pi 5 has been available for a little while now. We look at its advantages compared to the venerable Pi 4B and some other competing SBCs.

Max's Cool Beans by Max The Magnificent 36
Weird & Wonderful Arduino Projects, part 5: creating & optimising display algorithms.

Adding Solar Charging to a Renault Kangoo EV by Roderick Boswell 54
Roof-mounted solar panels and an onboard charging system can provide around 10,000 miles of free driving per year if the weather is good.

Precision Electronics, part five by Andrew Levido 60
Besides unwanted currents and fixed voltages, there is another type of unwanted signal that can cause all sorts of problems that we refer to as noise. So, how can we quantify its effects on our circuits and reduce the resulting errors?

Audio Out by Jake Rothman 66
An audio amplifier circuit that can switch between Class-A and Class-(A)B modes.

The Fox Report by Barry Fox 69
The little-known portal shutdown problem increasingly affecting Internet radios.

The History of Electronics, part five by Dr David Maddison 70
We wouldn't have computers, smartphones or AI without the work of thousands of people who created the foundations of modern electronics. This fifth article in the series covers inventions by companies and organisations up to 1962.

Regulars and Services

Editorial 2
Ferrite beads are not inductors

Reader services – Editorial and Advertising Departments 2

Subscribe to Practical Electronics and save money 59

Practical Electronics PCB Service 76
PCBs for Practical Electronics projects

Classified ads and Advertiser index 79

Next month! – highlights of our next issue of Practical Electronics 80

Made in Australia & the UK.

Written in Britain, Australia, the USA and Ireland.

Read everywhere.

© Silicon Chip Publications Pty Ltd 2025

Copyright in all drawings, photographs, articles, technical designs, software and intellectual property published in *Practical Electronics* is fully protected, and reproduction or imitation in whole or in part are expressly forbidden.

The June 2025 issue of *Practical Electronics* will be published on Thursday, 1st of May 2025 (see p80).

Renewals paid by PayPal, Visa and Mastercard are now handled via our website: www.electronpublishing.com

Practical Electronics

Volume 54. No. 5
May 2025
ISSN 2632 573X

Editorial

Editorial offices
Electron Publishing (Aust.) **Tel** +61 2 9939 3295
Unit 1, 234 Harbord Road **Email** pe@pemag.au
Brookvale NSW 2100 **Web** www.electronpublishing.com
Australia

Address mail to:
Electron Publishing (Australia)
PO Box 194, Matraville NSW 2036
Australia

Advertising enquiries
+61 2 9939 3295 pe@pemag.au

Editor	Nicholas Vinen
Publisher	Nicholas Vinen
Digital subscriptions	Stewart Kearn **Tel** 07918 614662
Online Editor	Alan Winstanley
Web Systems	Kris Thain
Production	Bao Smith
Technical staff	Tim Blythman, John Clarke

Print subscriptions
Practical Electronics Subscriptions
PO Box 6337
Bournemouth BH1 9EH **Tel** 01202 087631
United Kingdom **Email** pesubs@selectps.com

Technical enquiries
We regret that technical enquiries cannot be answered over the telephone. We are unable to offer any advice on the use, purchase, repair or modification of commercial equipment or the incorporation or modification of designs published in the magazine.

Questions about articles or projects should be sent to the editor by email: pe@pemag.au

Projects and circuits
All reasonable precautions are taken to ensure that the advice and data given to readers is reliable. We cannot, however, guarantee it and we cannot accept legal responsibility for it.

Some projects and circuits published in *Practical Electronics* employ voltages that can be lethal. Do not build, test, modify or fix any mains-powered equipment unless you fully understand the safety aspects involved and you use an RCD (GFCI) adaptor.

Component supplies
Silicon Chip Publications may offer kits or other parts for making our projects, but not in all cases. When kits are not available, readers will need to find and source parts themselves. We advise readers to check that all parts are still available before commencing any project in a back-dated issue.

Advertisements
Although the proprietors and staff of *Practical Electronics* take reasonable precautions to protect the interests of readers by ensuring as far as practicable that advertisements are bona fide, the magazine and its publishers cannot give any undertakings in respect of statements or claims made by advertisers, whether these advertisements are printed as part of the magazine, or in inserts. The Publishers regret that under no circumstances will the magazine accept liability for non-receipt of goods ordered, or for late delivery, or for faults in manufacture.

Transmitters/bugs/telephone equipment
We advise readers that certain items of radio transmitting and telephone equipment which may be advertised in our pages cannot be legally used in the UK. Readers should check the law before buying any transmitting or telephone equipment, as a fine, confiscation of equipment and/or imprisonment can result from illegal use or ownership. The laws vary from country to country; readers should check local laws.

Ferrite beads are not inductors

I often see ferrite beads drawn in circuit diagrams as if they are inductors, with “Lx” designators. While many circuit designers likely realise that they are not true inductors, treating them as such could cause confusion, especially for those reading the diagrams. This might lead them to assume that a ferrite bead is just another type of inductor, when in reality, it serves a very different purpose.

Ferrite beads exhibit some inductance – as do most components, including wires and PCB tracks – but their operation does not rely on it. At their simplest, ferrite beads are just a piece of wire passing close to (or through a hole in) a piece of ferrite.

Some of the confusion may stem from the fact that ferrite is used as a core material in high-frequency inductors and transformers. However, in those applications, the ferrite core is surrounded by multiple turns of wire to create significant inductance. A ferrite bead has just one or a few turns and thus a relatively low inductance.

Ferrite itself is a ceramic material that contains iron oxide. Like other magnetic core materials, it provides a path for magnetic flux, but only up to a certain frequency. Beyond that, ferrite becomes highly ‘lossy’, converting much of the magnetic energy to heat.

Ferrite beads take advantage of this property to suppress unwanted high-frequency signals by dissipating their energy, effectively acting as a frequency-dependent resistor rather than an inductor. Unlike an inductor, a ferrite bead does not store energy or resonate. It simply increases its resistance in a targeted frequency range to block unwanted signals.

So, rather than thinking of ferrite beads as inductors, it’s more accurate to consider them as a lossy impedance element that selectively dampens high-frequency signals. That distinction matters in terms of circuit design and how we draw them.

New telephone number for digital subs & shop items

Please note the new telephone number (shown opposite in red). You will have a better chance of getting a hold of Stewart via this new number. The number for print subscriptions (via Select Publisher Services) has not changed.

Nicholas Vinen, Electron Publishing (Australia)*
Publisher & Editor,
Practical Electronics Magazine

* a division of Silicon Chip Publications Pty Ltd.

Technological tidbits

Techno Talk

Max the Magnificent

Since I offer freelance technical consulting and writing services ("I write for food"), I end up chatting with a lot of companies about the latest and greatest technological offerings that are coming soon.

As I may have mentioned before on occasion, I am incredibly lucky. I'm not referring to my awesome looks, ready wit, or internationally-commented-on sense of fashion ("you aren't going out dressed like that, are you?", says my mother back in the UK). I mean that I get to chat with a cornucopia of companies on the cutting edge of technology, spanning myriad applications and industries. Let me share a few examples.

Feel the power

Just a couple of minutes ago, I was chatting with the folks at AmberSemi (https://ambersi.com). This small, fabless (and fabulous!) semiconductor company is poised to upend the power supply market. I'm talking about the AC-to-DC power supplies that plug in the wall and power our electronic doodads, gizmos, thingamabobs and what have you.

A common architecture for low-power AC-to-DC converters employs 1950s-era electromagnetic components like bridge rectifiers, transformers and high-voltage bulk capacitors. The folks at AmberSemi have replaced all of this with a solid-state semiconductor implementation that is much smaller and lighter, more efficient, more reliable, and cheaper.

Be the power

Speaking of power, a few days ago I had a call with the guys and gals at Circuit Mind (https://circuitmind.io). When they first came to my attention a year or so ago, they told me about their deterministic AI that helps electronic teams go "from architecture to schematic in 60 seconds".

You can use their interface to capture a high-level block diagram (microcontroller, temperature sensor, LCD display etc), annotate each block with any lower-level requirements (8-bit, 1°C accuracy, 640×480 resolution etc), while leaving anything you aren't sure about or don't care about unsaid. You use sliders to specify trade-offs like cost, area, power and component availability.

When you click the "Go" button, their system selects components, generates the circuit diagram(s) and provides a bill-of-materials (BOM), amongst other things.

In our most recent conversation, they informed me that they've just added a new power supply design module. This performs all the tasks discussed above for the power supply portion of the design.

People who can design power supplies (well, good power supplies) are few and far between, so this capability will be worth its weight in gold for many teams developing embedded systems and the "things" part of the IoT.

Losing count

Reticles, also known as photomasks, are critical components in the production of silicon chips. The high-end (size-wise) for today's biggest and baddest silicon chips is typically around 625mm^2 (25 × 25mm). The absolute limit is around 800mm^2 (28 × 28mm).

Recently, I was chatting with the lads and lasses at Synopsys (https://synopsys.com), a company that makes, among other things, widely used IC design software. They were excited to give me their predictions for chiplets and multi-die systems in 2025. As part of this, we touched on the new Blackwell graphics processing unit (GPU) from Nvidia.

There are two main flavours: the B100 and the B200. The latter consists of two 800mm^2 silicon dies joined by a 10 terabytes-per-second (TBps) chip-to-chip interconnect, forming a unified single GPU. In the same package, the B200 also contains 12 high-bandwidth memory (HBM) stacks, each of which contains multiple DRAM dies.

Excluding the HBMs, the B200 boasts 208 billion transistors (104 billion for each main die). How much power does this bodacious beauty consume? I'm scared to think, but…

Strike while the die is hot

A few days ago, I had an interesting conversation with the chaps and chapesses at Saras Micro Devices (https://sarasmicro.com). We started by looking at an image of an H100-based accelerator card that plugs into a server motherboard in a data centre, where the H100 (Hopper) GPU is the 2022/2023 predecessor to the Blackwell.

The H100 is a big package that occupies about ¼ of the accelerator card's real estate. The remaining ¾ of the card is taken up with power conversion and delivery components. 48V DC coming in is converted to 12V, which is subsequently stepped down to the point-of-load voltages. I'm told we are looking at 1000W @ 1V, which equates to 1000A.

In the not-so-distant future, when core voltages drop it 0.7V, we'll be looking at 1500A! Think about this; 1500A on a single, relatively small accelerator card, with 2000A heading our way. This is known as 'horizontal power delivery', because the power travels horizontally across the board from the power conversion components to the main processor.

This is where the folks at Saras leap onto the stage with a fanfare of flugelhorns (once heard, never forgotten). Their goal is to remove the inductors and capacitors that consume so much valuable real estate, and to embed these components in the board or package substrates.

This will pave the way for 'vertical power delivery', with much shorter tracks that will dramatically reduce IR voltage drops and associated I^{2}R thermal losses.

Memories are made of this

There are lots of different semiconductor memory technologies, the main contenders being flash, which is relatively slow but non-volatile (it remembers its contents when power is removed from the system); dynamic RAM (DRAM), which is cheap, has high capacity, and consumes little power; and static RAM (SRAM), which is expensive, has a low capacity, and is fast, but consumes lots of power.

When designing custom silicon chips like system-on-chip (SoC) devices, on-chip memory like caches and blocks of memory are usually made of SRAM to provide the highest possible performance.

Other memory technologies, such as magnetoresistive RAM (MRAM), have attractive qualities, like non-volatility and low power consumption, but they are significantly slower than SRAM.

I was recently conversing with the folks at Numem (https://numem.com). They have created an MRAM-based memory technology called NuRAM with a 'SmartMem' subsystem that results in MRAM with SRAM-like performance.

The scary thing is that I talked with all the companies mentioned here in just the past couple of weeks. Who knows what we'll learn in the days and weeks to come? PE

A short history of...

by Chris Morris

SINCLAIR CALCULATORS

Sinclair's innovative scientific and programmable calculators were all designed and assembled at Enderby's Mill, St. Ives, Huntingdon by Sinclair Radionics.

Consultant Nigel Searle developed the pioneering firmware for the first scientific calculator in Texas in 1973.

In March 1974, the first Sinclair Scientific went into full production. This calculator used the diminutive but elegant 1973 Cambridge case, in white with just 18 keys. It was nicely colour coded with no key lettering to wear off, other than on the arithmetic operators.

Selling in large numbers, it put Sinclair on a roller-coaster ride at a time when the new scientific calculator market was very dynamic.

The internal microcontroller was based on the pioneering Texas Instruments 'calculator-on-a-chip' (also used as the TMS0803 in the four-function TI Datamath). That sleek Datamath case, styled by Fred Gore & Associates of Carrolton, Texas, was meant to compete with the popular Bowmar 'Brain' - which had the largest sales in the world at the time for four-function calculators.

Incorporating compact mathematical algorithms suggested by Clive Sinclair (and probably Marvin Minsky at MIT), Nigel Searle reprogrammed the TI chip firmware in late 1973 to cram a dozen scientific and arithmetic functions into the same Datamath four-function ROM, using the sparse 320 micro-instructions available.

Reverse Polish Notation (RPN) was chosen for optimum efficiency, along with a two-level stack and the bare three registers on-chip. With RPN, rather than entering, say, 3 + 5 × 7 = to get 38, you would input 5, E, 7, ×, 3, +. The operators (× and + here) work on the last two entered or calculated values.

Ken Shirriff reverse-engineered that Sinclair/TI chip in 2013 and produced an ingenious simulator of a very clever calculator.

The Scientific could be slow and was accurate to only three decimal places at the margins, but it did the job at a very good price for the time (£50 on introduction, later falling rapidly in mass-production). The reprogrammed 28-pin PDIP PMOS chip (running from -10V and -16V supply rails) was named the TMC0805 by TI for Sinclair.

Power came from four AAA cells that had a 25 hour operating life. It used an excellent Bowmar Optostic nine-digit miniature LED display

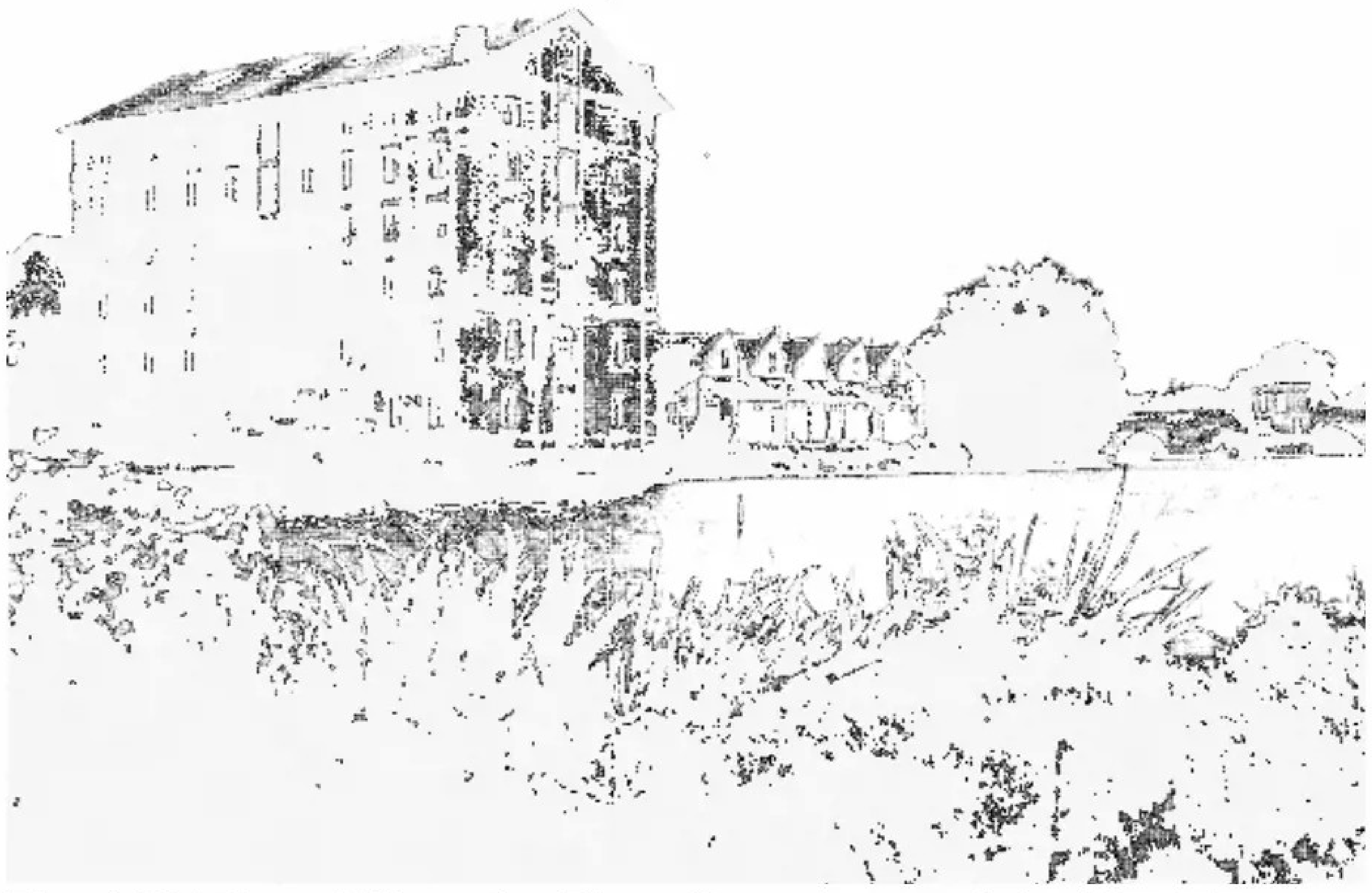
Originally, Bowmar had hoped to develop the Scientific chip for Sinclair as well, but they were pre-empted by Texas Instruments. Bowmar, started by Ed White in Boston, was the first manufacturer to successfully mass-produce the pocket calculator in North America. The sturdy Bowmar 901B Brain used the TI TMS0103 'calculator-on-a-chip' in September 1971.

Their LED/calculator factory, in Acton, Massachusetts, was later supplemented by Bowmar factories in Ottawa, Canada and Nogales, Mexico. Demand for both pocket calculators and LED displays grew at a dizzying rate between 1972 and 1974. Clive and Ed were good friends, and Clive bought hundreds of thousands of Bowmar LED displays.

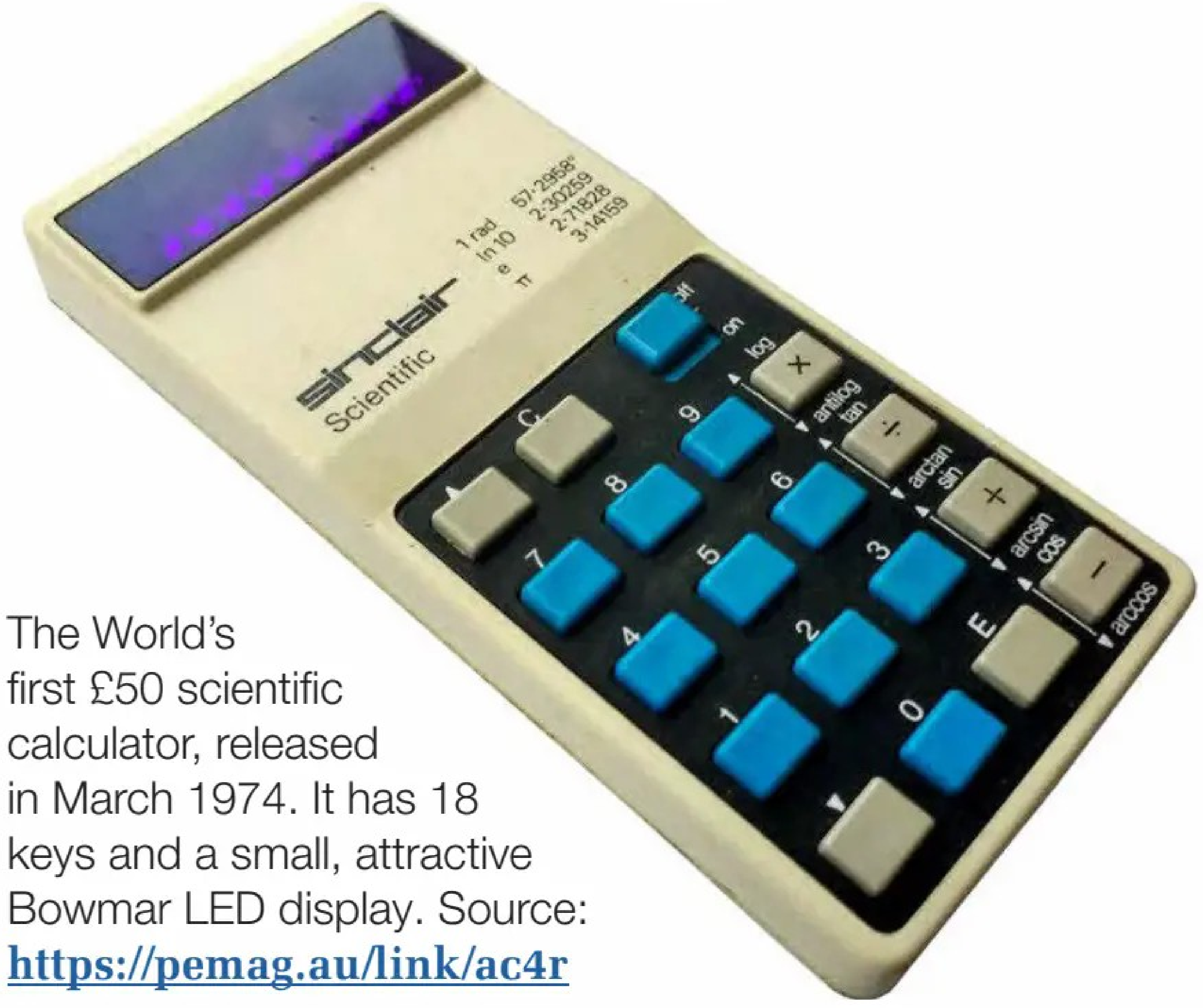
As it turned out, their calculators both shared the same pioneering LSI chip supplier, Texas Instruments. This became a mixed blessing later when, impressed by Bowmar's success, TI started making its own Datamath calculators using the same chip series from their own factory.

The 12 functions found on the Sinclair Scientific were the usual four arithmetic ones (+, -, × & ÷) plus sine, cosine, tangent, arcsine, arccosine, arctangent, log and antilog (exponentiation). The calculator worked in radians, and scientific notation (eg, 1.23E4 = 12,340) was used to conserve instructions, with six-digit precision. It became a best-seller in its time (1974-76).

Most ingeniously, the key constants for deriving yet more functions were



The 1854 Corn Mill on the River Ouse, home of the Sinclair Radionics Factory.



The World's first £50 scientific calculator, released in March 1974. It has 18 keys and a small, attractive Bowmar LED display. Source: https://pemag.au/link/ac4r printed on the case below the display, to six figures, as there was no room left in the hard-pressed ROM. This enabled the user to also derive x^y, e^x, $\sqrt{x}$ and π. This was well-explained in the comprehensive 32-page manual.

More recently, the original Sinclair Scientific's later 'super-hacking' popularity has resulted in several modern 'emulated' versions being produced in both the USA and Canada. They use Ken Shirriff's Sinclair retro firmware that includes the original 320 word microcode, though it has been moved to a modern Atmel AVR microcontroller.

The next big thing

The next development at Sinclair came in a larger and restyled case. This was developed on contract for Gillette as a simple four-function calculator called the GPA. Sinclair reused the case for their new Oxford series, while Gillette decided to stick to razors.

This initially resulted in the Oxford 300 scientific, introduced a year after the pioneering Scientific, in March 1975. However, it used a single scientific chip (the CF5966) that General Instrument Corporation developed later. The chip was not unique to Sinclair and, in fact, was widely used by other competitors.

Trig and log results were slightly more accurate than before, to the fourth decimal place instead of the third. The Oxford case, much bulkier and heavier than the slim Sinclair Scientific, had just 19 large keys.

It was powered by a 9V PP3 battery, driving a green vacuum fluorescent display (VFD) from Noritake Itron, powered by -24V derived using an Astec DC-DC converter module (which also provided the -16V for the GIC PMOS chip).

VFDs had been developed in the early 1970s by the Japanese calculator industry, as they were initially barred by tariffs and patents from obtaining LED displays from the USA. LEDs used simpler low-voltage circuitry and drew less current than VFDs.

The Oxford case snapped together with plastic rivets; no screws were used. The thin main circuit board required unexpected bracing inside in some cases (with glued wood offcuts on my original factory example!), indicating a rushed, post-Gillette GPA development of the Oxford design.

Despite the heavy-duty outer case, the 'bendy' main circuit board inside was only 0.7mm thick; that's precariously thin, around half the 'standard' PCB thickness of 1.5-1.6mm. By comparison, the pioneering Sinclair Scientific had a sturdy 1.6mm-thick single circuit board.

Radians or degrees were selected by a slide switch on the case's upper end. The Oxford does not seem to have been as popular as the Cambridge series, and was never available as a kit (it was probably too awkward to assemble). Cambridge Scientific mail-order kits were very much in favour amongst Heathkit enthusiasts in the USA, according to Nigel Searle in a 2021 YouTube interview.

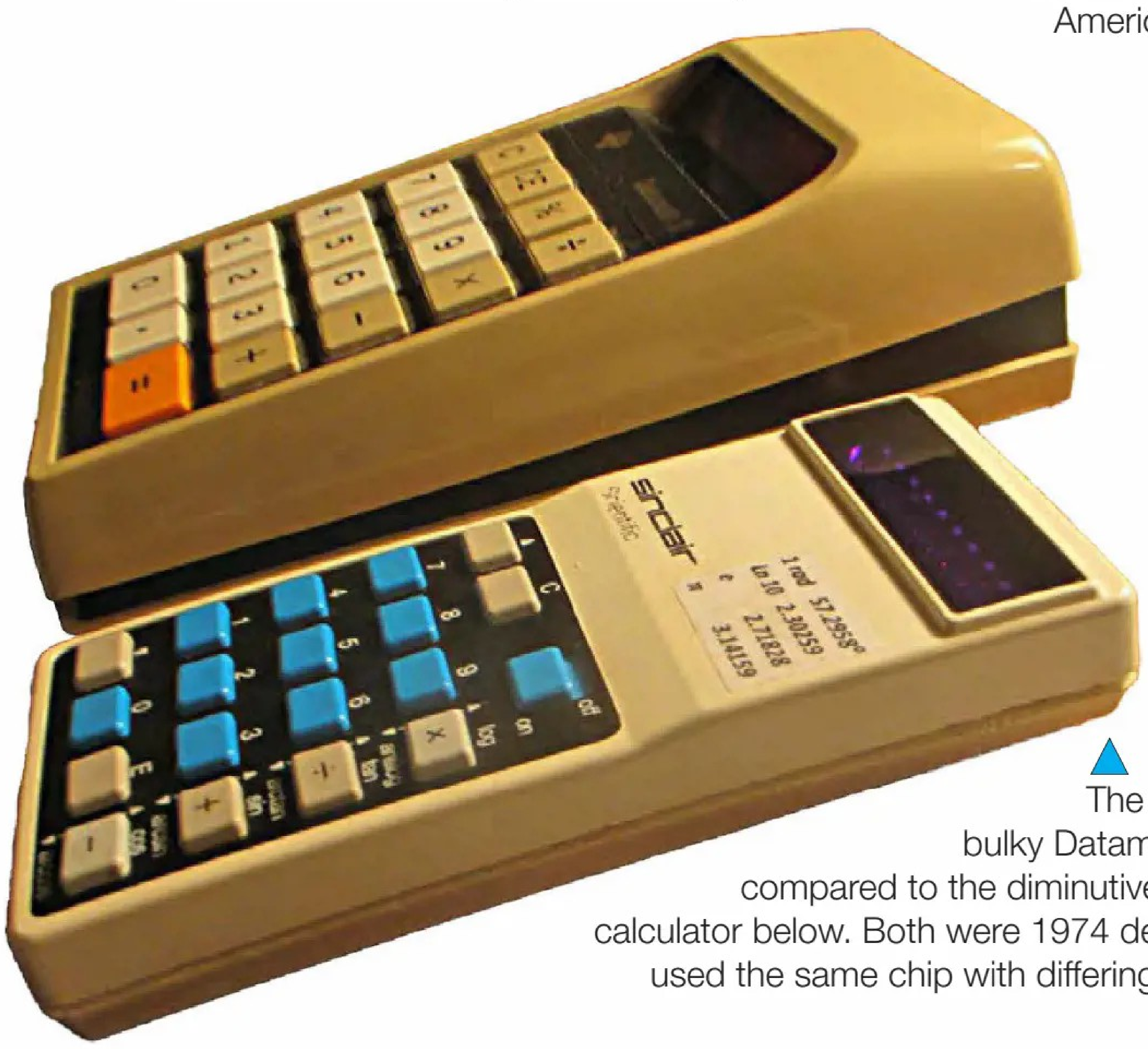
In retrospect, the Oxford looks large and clunky even now, when compared to the best-selling Hewlett Packard HP-25 programmable of that year, never mind the

Gillette GPA ™
K
C
CE
7
8
9
÷
4
5
6
×
1
2
3
−
0
•
=
+

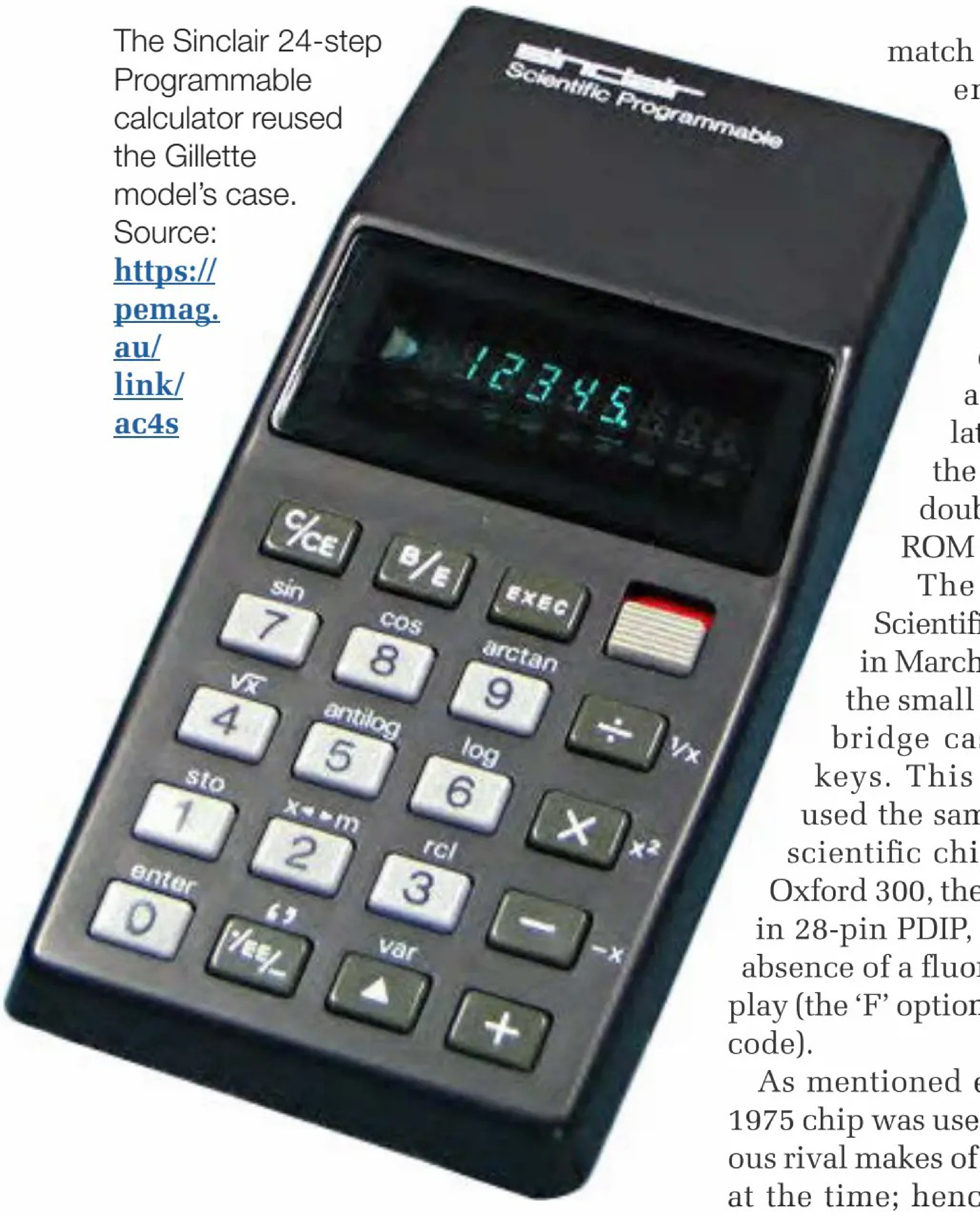
The contract Gillette GPA calculator. The same case was used in the first Sinclair 24-step Programmable calculator.

BOWMAR
ON
K
D
CE
C
7
8
9
−=
4
5
6
÷
1
2
3
×
0
•
+=

The very first US pocket calculator, the Bowmar 'Brain' 901C, priced at $240 in September 1971. Mine was made in Ottawa, Canada. Usage instructions were on a label on the underside. Image source: National Museum of American History.



The stylish but bulky Datamath II (top) compared to the diminutive Scientific calculator below. Both were 1974 designs that used the same chip with differing firmware.



The Sinclair 24-step Programmable calculator reused the Gillette model's case. Source: **https://pemag.au/link/ac4s**

sleek Cambridge or later clever Sinclair Enterprise. Inside, the main circuit board/display and the keyboard PCB are rather crudely interconnected and supported. Having only 19 keys in such a large case seems like an opportunity missed.

Further improvements

In the following year (1976), the follow-up Sinclair Oxford Scientific came out, in a white case. This used the GIC CF596 chip instead, adding y^x, log(x), x^2, ± and double brackets. There were not enough keys left to use the 10^x and % functions also available on the chip. For the same reason, three of the five previous memory options on the Oxford 300 keyboard had to be dropped.

In August 1975, five months after the popular HP-25 programmable launched at US$200, the Sinclair Scientific Programmable was produced for £30, in the same black Oxford case as the 300, with a green VFD display but with RPN operation, using custom Sinclair firmware. This was a failure as a product, despite being only 40% of the HP cost.

With no branching, no step/debug, a single memory and 24 steps, it could not be considered a real programmable calculator, even at the time. It was more like the equally unsuccessful National Semiconductor "Novus Programmables" of 1975 in function, just a learn-mode machine.

Even as a 'scientific calculator' it didn't match the original 1974 Scientific, inexplicably lacking tan, arc sin and arc cos. The firmware was likely again programmed by Nigel Searle, who had been assigned by Clive Sinclair to rapidly develop a programmable calculator after the success of the Scientific, though undoubtedly with a too-small ROM and RAM.

The Sinclair Cambridge Scientific was released in March 1976, using the small white Cambridge case with 19 keys. This time, they used the same standard scientific chip as in the Oxford 300, the GIC CF596 in 28-pin PDIP, despite the absence of a fluorescent display (the 'F' option in the chip code).

As mentioned earlier, that 1975 chip was used in numerous rival makes of calculators at the time; hence, Sinclair only had styling and price to differentiate themselves from a raft of competitors. It eschewed RPN for algebraic notation. Interestingly, only two AAA cells were used instead of four, saving weight and space, although reducing the running time with the heavier current draw.

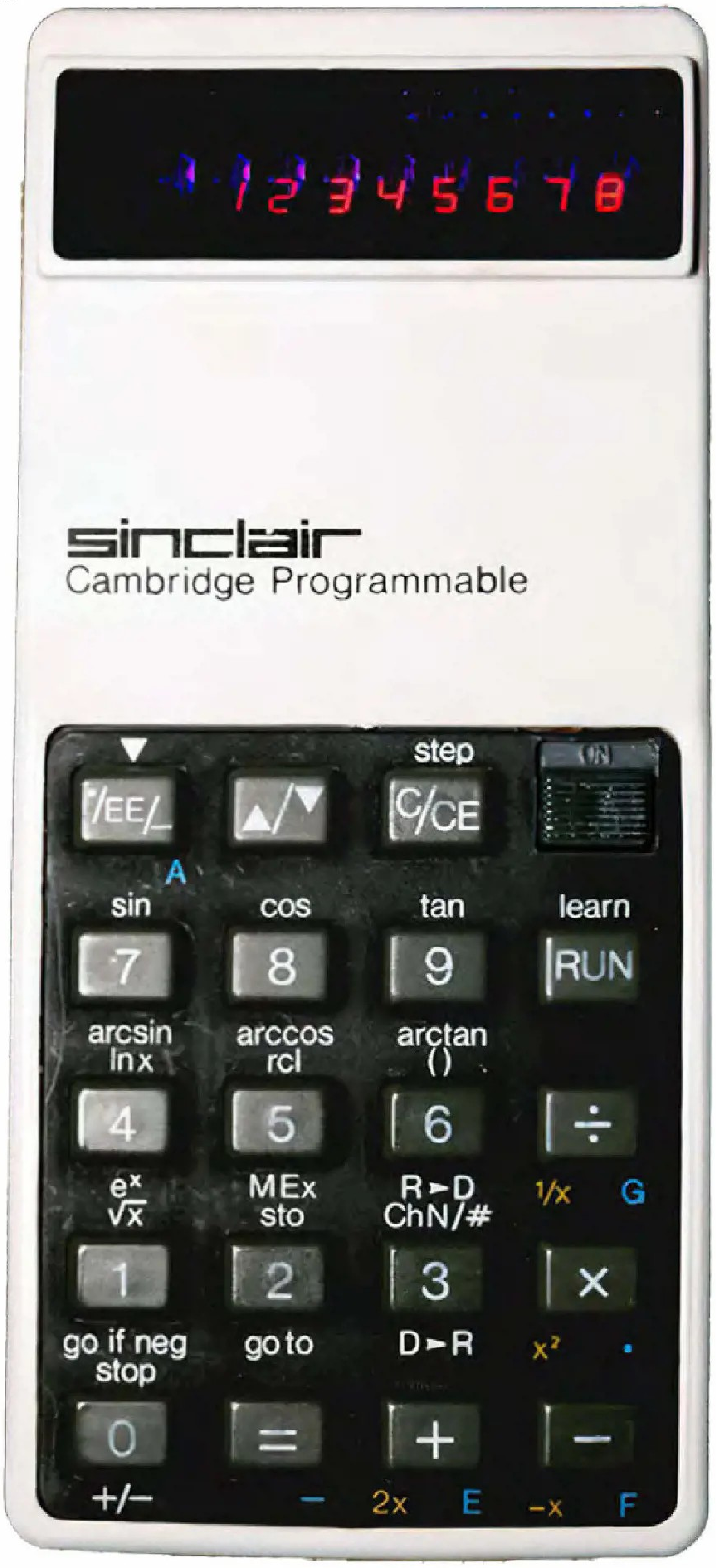
A radian/degree slide switch was fitted in the unused half of the battery compartment, requiring the battery lid removed in operation to change this trigonometric option. Compared to the original Scientific, natural log (ln)/e^x was substituted for log(x)/10^x, with 1/x, $\sqrt{x}$, π and a single memory slot added.

By early 1977, Sinclair was fitting the more advanced GIC CF5986 chip to this model instead, though there were no spare keys available to use the new y^x or log(x)/10^x functions. As with the original Scientific of 1974, the two Texas Instruments custom support chips (inverter and LED digit drive), IFC1 and IFC2, were also fitted.

One wonders why the fluorescent drive version of the chip was used when it was only driving low-voltage LED segments. Only those at the Mill knew.

The Sinclair Cambridge Programmable was introduced in September 1976. This was the first serious programmable calculator from Sinclair, although it came quite late during the fast-moving mid-1970s world of programmable calculator development. It came in the white Cambridge mini case. It had just 36 steps and many of the 19 keys had to perform triple duty.

With only a single memory register, it was not very effective, although it did have go-to and go-if-negative branching, plus a crude step/debug display. Algebraic logic was used, based on custom



The 36-step Sinclair Cambridge Programmable calculator, introduced in September 1976. It wasn't especially successful. Image source: Kevan Dickin (**https://pemag.au/link/ac4t**).

Sinclair programming of a National Semiconductor MM5799 COPS microcontroller in a 28-pin PDIP package.

The MM5799 chip had 1536 microinstructions in its ROM, plus 384 bits of RAM for the 36-step program and a single memory slot. A DS8874 shift register was used to drive the nine-digit LED display. This display was also from National Semiconductor.

Bowmar (the biggest calculator manufacturer by volume in the world in 1973) had gone bankrupt by February 1975, at a time when calculator prices had fallen to 10% of their 1972 levels.

Radio Shack sold a re-badged version of this same Sinclair Cambridge Programmable called the EC-4001. Both machines had an unsightly ‘camel hump’ 9V battery compartment at the back. This spoiled the sleek Cambridge look and prevented the calculator from sitting flat on a desk. It could no longer fit in the neat hardshell carrying case that had helped to define the Cambridge calculator family.

Only 16 op codes were displayed; many were used ambiguously for three different functions at a time. No fewer than seven steps were needed to halt an accidental endless loop in a program; an example of the awkward programming model and limited number of keys.

The operating manual was carelessly written (Sinclair manuals were usually very good) – it didn’t even include an illustration of the keyboard! The programming section occupied only a third of the slim 25-page manual, compared to half of the later 45-page Enterprise manual.

As with the Cambridge Scientific, the lettering rapidly wore off the number keys (the original Scientific only needed lettered keys for the four arithmetic functions). Ironically, this was the one area where the Oxford series were superior; they used the Hewlett Packard method of double injection moulded keys, with two plastic colours in each key, so the lettering could never fade.

However, this more expensive feature was dropped by 1976, as is apparent from the faded keys on the later Sinclair Oxford Scientifics from that year.

Finally, in July 1978, as Sinclair Radionics was in its final phase, the most successful Sinclair programmable calculator yet was produced. Available for the very reasonable price of £25, this was the Sinclair Enterprise Programmable. “Enterprise” referred to the new owners of last resort, the National Enterprise Board.

Housed in an elegant new slimline case, parting easily to access the 9V PP3 battery without a separate cover, the Enterprise had the go-to and go-if-negative branching instructions of the 1976 Cambridge Programmable, plus a respectable 7-entry storage register and memory arithmetic.

A much more usable op code display for debugging was provided (with 42 separate op codes). A substantial three-volume program library was available. Precision was excellent at eight digits. Despite a very capable program length of 79 semi-merged steps, subroutine calls were sadly not included.

Nevertheless, it was a good, reliable and practical machine intelligently laid out, showing the same innovative spirit as the 1974 Scientific.

As with the earlier 1976 programmable, a National Semiconductor 4-bit COPS PMOS family chip was again used (this time, the MM57146 with the MM57126 serial RAM for storing the 79 program steps) with custom Sinclair firmware, but to much better effect.

The keypad had a good ‘click’ feel (as indeed did most Sinclair calculators with their Klixon-style action). The expanded set of 24 keys (versus 18 or 19 keys in all previous Sinclair machines) was much easier to use and reduced the separate steps needed in a program.

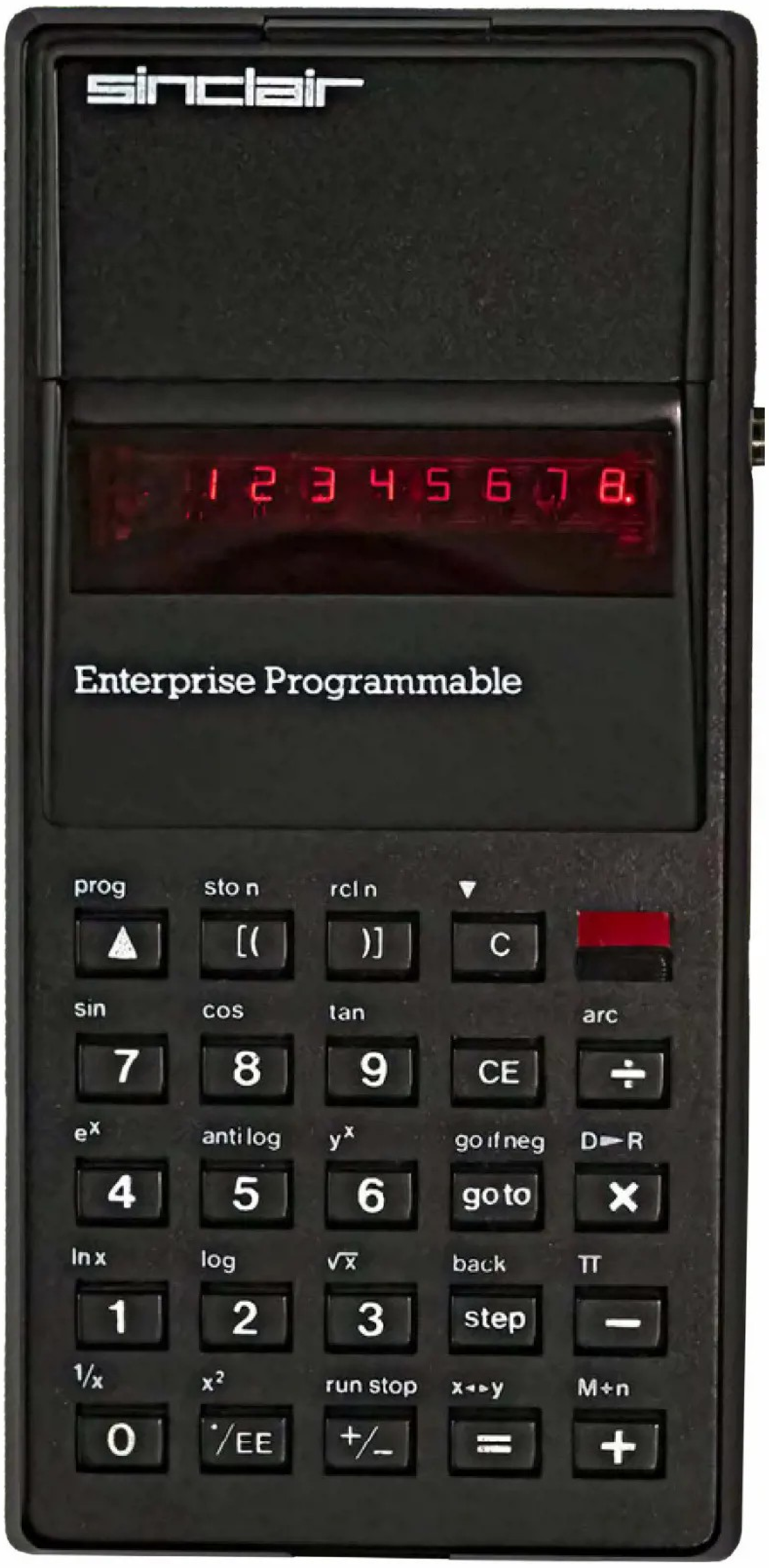
A more durable key printing process ensured the key lettering did not rapidly fade, as had happened with the earlier Cambridge models (which had used poor contrast grey keys with fragile white lettering).

In a sense, Sinclair’s scientific and programmable calculators came (1974) and went (1978) with a bang, incorporating excitement, innovative design and styling with good prices. The years in between, not so much.

The world-wide evolution of the scientific calculator in the 1970s was unbelievably fast. Sinclair played a significant role during that energetic time, bringing both an elegant appearance to an everyday object and (sometimes) very clever firmware and operational design, with well-written manuals and program libraries.

Their calculators were widely exported, often as kits and through mail order, particularly to the United States via their New York City office on East 57th Street. To this day, the 1970s Sinclair calculator family is regarded as iconic, both in England and North America.

In July 1979, Clive Sinclair left the plush top floor of the Mill and Sinclair Radionics, never to return. His next successful endeavour, the ZX80/81 and Spectrum personal computers, began from Kings Parade, Cambridge with a new company, Sinclair Research. But that is another story…



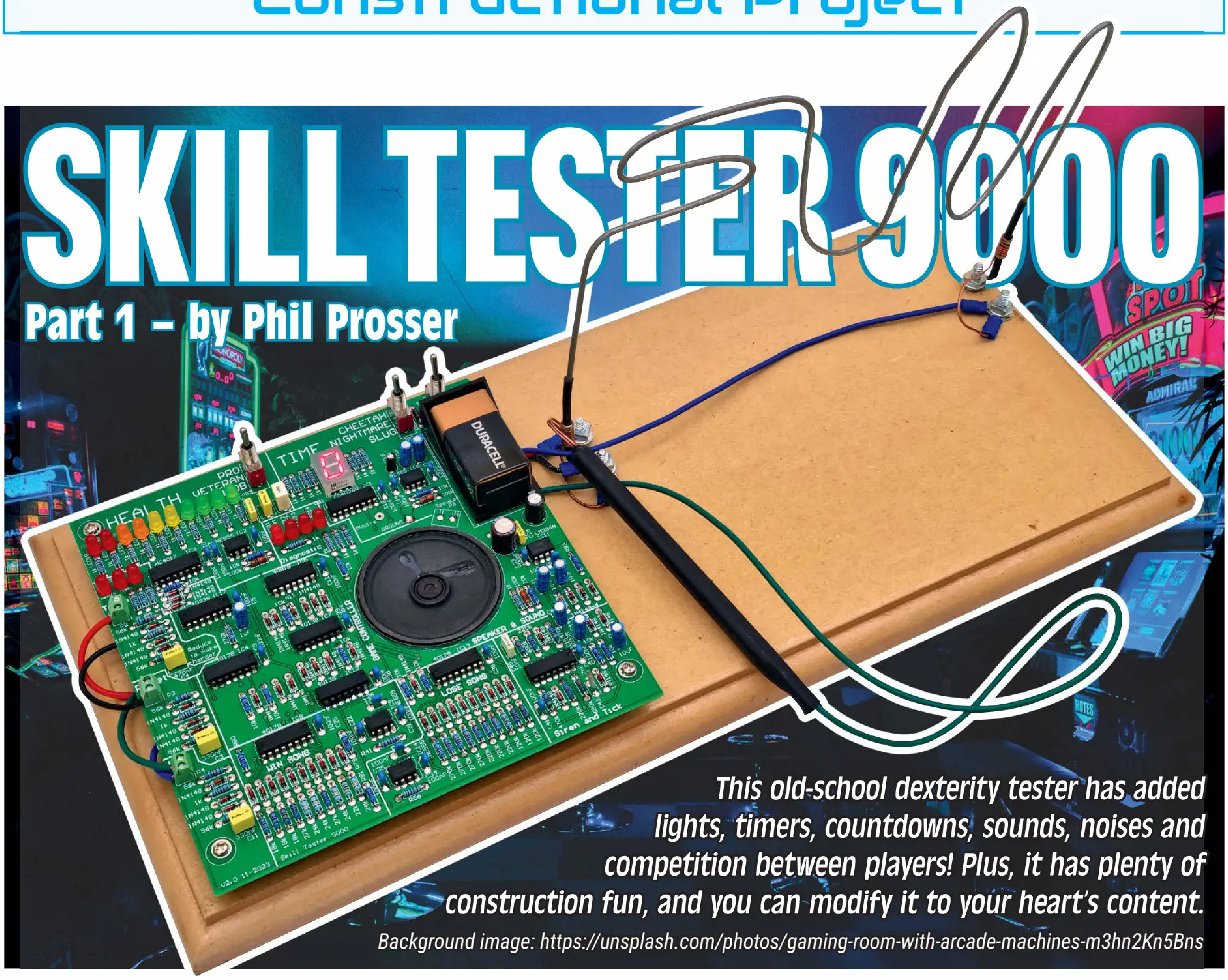
The third, successful Sinclair Programmable calculator had 24 keys in a slim, attractive case and easy battery access (by lifting off the top cover). Source: Kevan Dickin.

References

- http://files.righto.com/calculator/sinclair_scientific_simulator.html
- https://hackaday.io/project/91895-sinclair-scientific-calculator-emulator
- https://www.tindie.com/products/arduinoenigma/sinclair-scientific-calculator-emulator/
- https://theretrohour.com/sinclair-inside-story-nigel-searle-ep256/
- https://www.wass.net/othermanuals/ (look for General Instruments Micro-Electronics Calculator Chips Catalog datasheets 1978)
- https://spectrum.ieee.org/the-consumer-electronics-hall-of-fame-bowmar-901b
- https://www.rsp-italy.it/Electronics/Databooks/National%20Semiconductor/index.htm (look for 1977 MOS LSI data book, pp9-27) PE

SKILL TESTER 9000

Part 1 – by Phil Prosser



This old-school dexterity tester has added lights, timers, countdowns, sounds, noises and competition between players! Plus, it has plenty of construction fun, and you can modify it to your heart's content.

Background image: https://unsplash.com/photos/gaming-room-with-arcade-machines-m3hn2Kn5Bns

This project reinvents that simple and fun game of skill where you need to navigate a loop of wire along a convoluted path without sounding a buzzer. That old game had no clear 'win or lose' scenario, nor did it add competitive factors such as time limits or measuring your speed against your friends.

The buzzer version is easy to design, but how about we add more sounds than just the buzzer, making it more exciting to play? Enter the Skill Tester 9000!

This project is all about fun mixed with a bit of learning. Younger builders can just solder parts to the PCB to get a working game, while more mature constructors can look into how the logic works and change the sounds by varying resistor and capacitor values.

When considering how to design this game, the obvious answer in 2025 is to grab a microcontroller and write everything in software, including the game logic and sounds. That would result in a board with just a handful of parts and a loudspeaker, which would be small and cheap. The problem is that it is not that much fun, and constructors cannot easily tweak any aspect of the project.

There's also relatively little to learn from such a design unless you're willing to delve into the source code.

Responding to feedback for 'less micro stuck on a board' projects and seeking to give builders a test bed on which they experiment with some old-fashioned discrete logic, we decided to stick to commonly available 4000-series logic chips and discrete through-hole components.

While there are a lot of parts to solder, it is easy to build overall and delivers that therapy of soldering a bunch of parts to a board. We also think the result is pretty cool in an old-school way. Given this implementation, there is little in this project that you cannot tweak. Maybe it is just me, but I find that fun.

As can be seen from the photos, the Skill Tester 9000 has a complicated wire 'maze' that you need to run a hoop along without touching. We have just added a bunch of technology to make it more fun and competitive. When building it, you need to decide how dastardly you make the wire maze, which affects the difficulty factor.

It's powered by a 9V battery that gives decent runtime and avoids the need for plugpacks and the like. It is, of course, possible to use a 9V DC mains supply, and all the parts on the board can operate over a wide voltage range, so there is no need for regulation if you take that route.

Given that this project is entirely made from parts that have been available for around 50 years, the following question came to mind: why hasn't it been done already? I think the answer lies in the cost of materials, especially the PCB. This project would be impractical without a double-sided PCB and moderately thin traces. As recently as 10 years ago, the PCB cost would have been prohibitive.

Designing the game

Watching youngsters play modern games, a few themes became apparent.

- The games are competitive.
- They often incorporate difficulty levels.

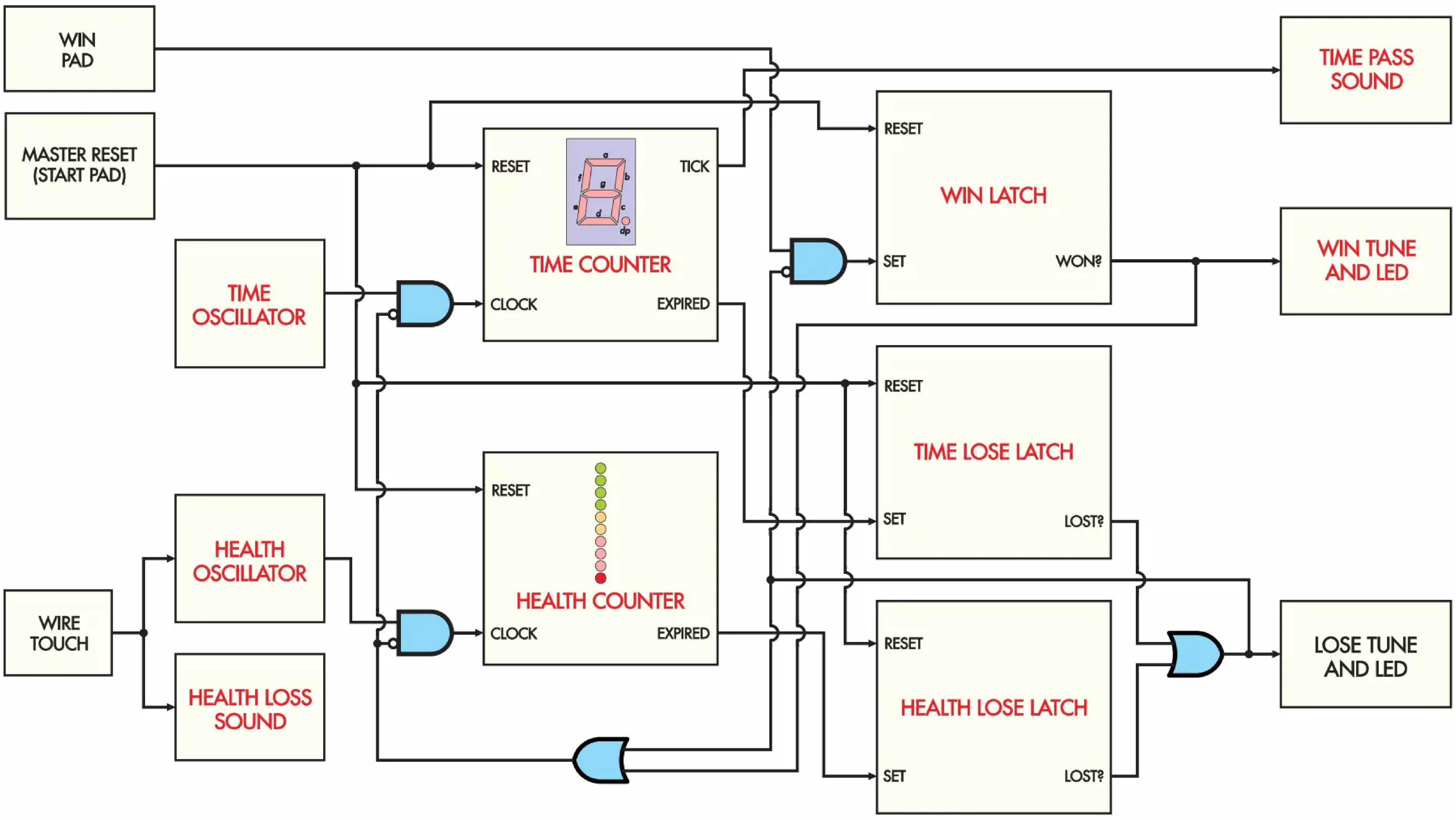


Fig.1: this simplified version of the game logic shows how it broadly works. The game starts after the circuit is reset and is won if the Win Pad is touched before either Lose condition is met (out of Time or out of Health from touching the wire). Sounds are produced for each time tick, if the wire is touched (and Health is lost), if the game is won and if the game is lost.

- Characters 'take hits' and lose health; if this runs out, they lose.
- The games have a sense of urgency, often in the form of time limits.
- Sound plays a big role; we want to hear things like time passing, an alarm if the wire is touched, a distinctive tune for winning and a depressing tune for running out of time or losing all your health.

We decided to use 4000-series CMOS logic to implement these functions, which is both cheap and widely available. We can do that as follows.

Health is a commodity that starts full and is reduced each time the player touches the wire. A logic block must detect if the wire is being touched and determine the duration. We do this using a 4017 decade counter that we can 'clock' at a slow, medium or fast rate to implement three difficulty levels. That determines how long you can touch the wire before you run out of health and lose.

To win, the player must navigate the course within a set Time. This is implemented in a logic block comprising a clock source, a 4026 digital counter and a 7-segment decoder. The clock speed for this counter can also be varied to determine how quickly you need to traverse the game to win.

Winning is pretty important. We have added a pad at the end of the course that the player must touch. This stops the timer and health counter, and if you have health and time left, it will play a victory song.

The majority of the circuit components are to track health and time, determine the winning and losing conditions, and play the various sounds. Sounds are triggered if health is reduced as time passes and if the player wins or loses. While these parts all interact, they can be analysed as standalone blocks.

Melodic sounds are better than a simple buzzer. We have taken a couple of approaches here, illustrating a few concepts we have seen over the years. The 'touch' sound uses a couple of logic-based oscillators to make a two-tone siren noise. This is implemented using Schmitt-trigger NAND gates and a few discrete parts. We have tweaked this to make it a nasty, alarming sound.

The Tick sound for time passing is derived from the overall game timer and uses a similar siren circuit, but adds a very simple circuit to 'shape' the sound into a fast attack and slow decay. We have tweaked this to make the tick less of an alarm but still add urgency to the game.

For the Win sound, we have used a circuit that allows us to play 10 notes, each at an independent frequency. This amounts to a clock circuit that defines each note length and a 4017 decade counter that changes the resistance in a 555 astable oscillator circuit.

After each clock pulse, the resistor in series with the respective 4017 output sets the frequency of the 555, allowing us to program a 10-note tune by choosing those resistor values.

The Lose tune is precisely the same circuit as Win, but we have set the resistors to make a sad tune rather than a happy one. Those of you with more musical sense than us may disagree with the tunes we have set – all you need to do to make your own is fiddle with these resistor values! Adding a diode would allow you to change the tune length if you want to; we will leave that to you.

The resulting game logic is shown in the simplified block diagram, Fig.1. It performs the following tasks.

- When the game is reset, the Time counter starts running, and the Health counter is reset to full. It is ready to play.
- A ticking sound is made each time the Time counter is reduced.
- During the game, touching the wire decreases your health and makes a noise.

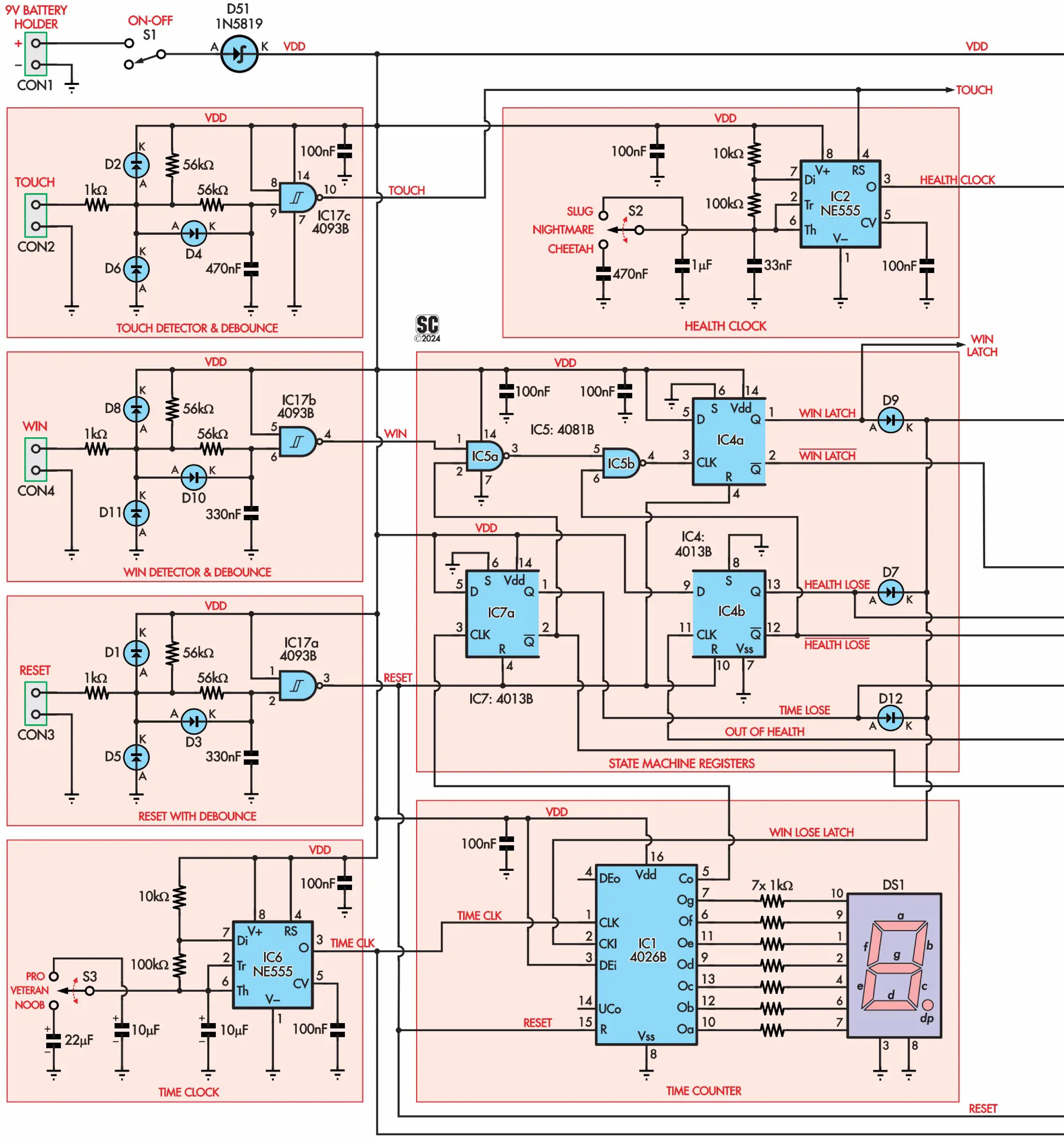
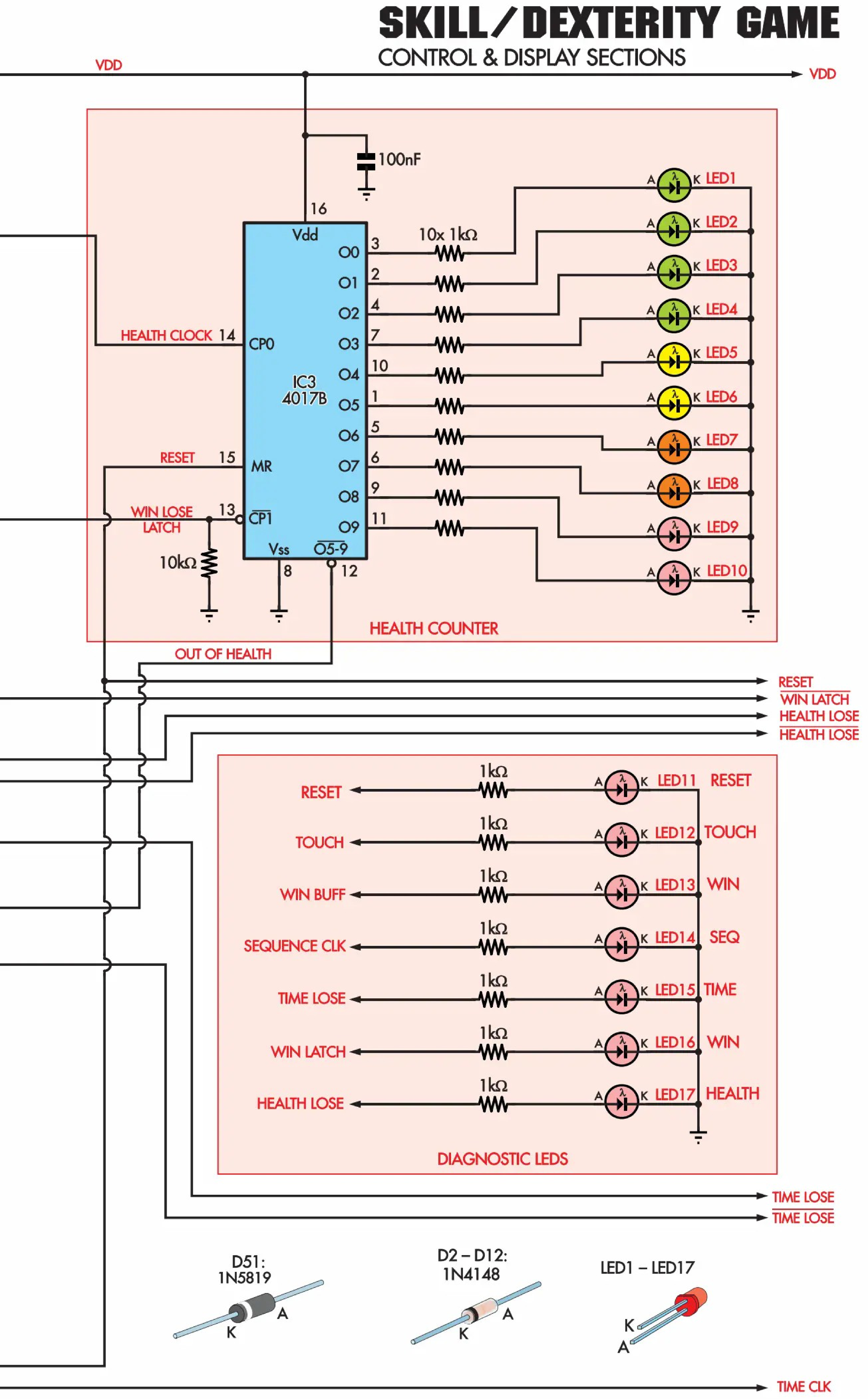

9V BATTERY HOLDER
CON1
ON-OFF S1
D51 1N5819
VDD
TOUCH
TOUCH DETECTOR & DEBOUNCE
CON2
D2
D6
D4
1kΩ
56kΩ
56kΩ
470nF
100nF
IC17c 4093B
HEALTH CLOCK
SLUG
NIGHTMARE
CHEETAH
S2
470nF
1μF
33nF
100nF
10kΩ
100kΩ
IC2 NE555
HEALTH CLOCK
SC ©2024
WIN
WIN DETECTOR & DEBOUNCE
CON4
D8
D11
D10
330nF
IC17b 4093B
WIN LATCH
STATE MACHINE REGISTERS
IC5: 4081B
IC5a
IC5b
IC4a
IC4: 4013B
IC4b
IC7a
IC7: 4013B
D9
D7
D12
HEALTH LOSE
TIME LOSE
OUT OF HEALTH
RESET
RESET WITH DEBOUNCE
CON3
D1
D5
D3
IC17a 4093B
TIME CLOCK
PRO
VETERAN
NOOB
S3
22μF
10μF
10μF
100nF
IC6 NE555
TIME CLK
TIME COUNTER
WIN LOSE LATCH
IC1 4026B
7x 1kΩ
DS1

Fig.2: this half of the circuit diagram includes all the game logic. Three identical debouncing sections are provided for each input (all at upper left), while the State Machine Registers section in the middle keeps track of the game state. The remaining sections implement the health counter, the time clock and some debugging LEDs.

- If you reach the wire's end and touch the Win pad before the time or health run out, the system goes into the Win state. It plays a happy song, and the Win LED latches on.
- If the time or health hit zero before you touch the Win pad, the system goes into the Lose state. It plays a sad song, and the Lose LED latches on. The Win state can no longer be triggered until a new game is started.
- In the Win or Lose state, the Time counter stops so you can see how fast you did it and the Health counter stops so you can see how much health you had left.
- Pressing Reset starts another game.

The Reset button is best as a pad at the start of the wire rather than a separate button. That way, you're ready to


SKILL/DEXTERITY GAME
CONTROL & DISPLAY SECTIONS
VDD
VDD
100nF
16
Vdd
IC3
4017B
10x 1kΩ
O0
O1
O2
O3
O4
O5
O6
O7
O8
O9
3
2
4
7
10
1
5
6
9
11
HEALTH CLOCK 14
CP0
RESET 15
MR
WIN LOSE LATCH
13
CP1
Vss
O5-9
8
12
10kΩ
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
LED9
LED10
A
K
HEALTH COUNTER
OUT OF HEALTH
RESET
WIN LATCH
HEALTH LOSE
HEALTH LOSE
1kΩ
RESET
TOUCH
WIN BUFF
SEQUENCE CLK
TIME LOSE
WIN LATCH
HEALTH LOSE
LED11 RESET
LED12 TOUCH
LED13 WIN
LED14 SEQ
LED15 TIME
LED16 WIN
LED17 HEALTH
DIAGNOSTIC LEDS
TIME LOSE
TIME LOSE
D51: 1N5819
D2 – D12: 1N4148
LED1 – LED17
K
A
TIME CLK

go as soon as you start the game.

Implementation

This is intended to be a fun project that allows people to build and play with one another, show off some old-school logic, and let people see how it works. Thus, the entire game is built on one PCB that houses the battery and speaker. It can be screwed to the board that holds the Skill Tester 9000 wire.

All solder pads have been made as large as practical and with good spacing so that younger people can build it successfully. There are quite a few bits, but you will note that, for example, all bar one of the diodes are the 1N4148 type that's dead easy to solder.

We have incorporated a lot of extra LEDs that show the system states so we can see each part of the circuit operate without the whole thing having to be complete and operational in one hit. That's especially good for those with shorter attention spans.

We have used a range of coloured LEDs on the health bar, starting with green and then going to yellow, orange and red as the Health bar runs down. After all, sound and colour communicate good and bad well, creating excitement, which matters in a game like this.

Circuit details

The game is controlled by three key latches: Win, Time Lose and Health Lose. If implemented in software, it could be done as a classic state machine. The states have been simplified to a reset state and the three win/lose states to keep the parts count manageable.

We use three D-type flip-flops to store the state of the game. After Reset has been asserted, the Win, Time Lose and Health Lose latches are all cleared to 0, and the game runs. The game continues until one of these latches is set; then, the game stops. While a flip-flop is technically not identical to a latch, they are similar, so we can consider them equivalent here.

IC4a, IC4b and IC7a are the flip-flops that store those states. These are 4013 D-Type flip-flops, part of the 4000 series of logic that came out in 1968, still widely available and used today.

A flip-flop stores a single bit of data, where the Q and $\overline{Q}$ outputs represent the value stored. ($\overline{Q}$ is the inverse of Q; 1 instead of 0 or 0 instead of 1). The bar over the name means it is active low, or the inverse of the plain signal name.

The device has data (D), set (S) and reset (R) inputs. The logic value present at the D input is stored in the flip-flop when the clock (CLK) signal transitions from low to high and then appears on the Q output. This only happens on the rising edge for most flip-flops, which is very important in digital design.

The fact that data is only latched on the clock rising edge allows digital designers to work out all the delays in their system to ensure that the D input level is stable before the clock edge, or else things would go haywire. The set and reset pins on these ICs allow these latches to be set to one or cleared to zero asynchronously (ignoring the clock input). That means these flip-flops can also act like latches.

Thankfully, our clock rates are 1-20Hz, about a billion times slower than your PC and a million times slower than the 4000 series logic can handle. However, the principle of latching and storing our few bits of data still applies.

Our control logic is shown in the "state machine registers" section of the first part of the circuit diagram, Fig.2. Yes, this game is truly asynchronous!

When the Reset line is high, all the latches are reset (the counters are also reset, but we'll get to that later).

When Reset goes low again, all three flip-flops have Q=0, and diodes D9, D7 and D12 'OR' these signals together, producing a 0 on the Win Lose Latch line, starting the game. That line remains low until one of the flip-flop Q outputs goes high, at which point Win Lose Latch goes high. That means the game ends, and the counters stop, whether the player won or lost.

The associated 4017 or 4026 counter IC will overflow if either time or health runs out. When this occurs, they have a carry-out (Co) pin that goes from low to high. That is connected to the clock input of our D-type flip-flop, which you will recall will clock the data on the D input to the Q output on the rising edge of the clock.

So, if you run out of health or time, the Health Lose or Time Lose signal will go high.

Our D-type flip-flops have a convenient $\overline{Q}$ inverted output, which is high when the game starts, and we can use AND gates to enable the Win input signal using IC5a and IC5b. When one of the Time Lose or Health Lose outputs goes high, the respective $\overline{Q}$ output goes low. That disables the input to the Win flip-flop, so you can no longer win the game until it is reset.

If the player touches the Win pad at the end of the course, that generates a high Win signal that is ANDed with $\overline{\text{Time Lose}}$ and then $\overline{\text{Health Lose}}$. The resulting signal drives the clock input of the Win flip-flop, causing its Q output (Win Latch) to go high.

Once one of the latches is triggered, the only way for the system to become active again is for Reset to be touched, which resets the system to its initial state.

Now that we know how the game control works, let's look at how the timers and sound generation work. Each section is quite self-contained and generally is either triggered by a state or enabled by an event such as a clock tick.

Sound generators

The sound-generating part of the circuit is shown in Fig.3 (see overleaf). Together with Fig.2, these two diagrams show the complete circuit of the game. That's except for the wand, wire, reset (start) and win pads, which connect to the terminals of CON2, CON3 & CON4; as shown in Fig.4.

The touch sound generator is a classic CMOS logic sound circuit using two oscillators. The pin 1 input of NAND gate IC15a is tied to the positive rail (logic high), so the gate acts as an inverter, with pin 2 being the input (we could get the same effect by tying the two input pins together).

The output goes back to the input through a resistor, and the input has a capacitor to ground, creating an RC (resistor/capacitor) oscillator.

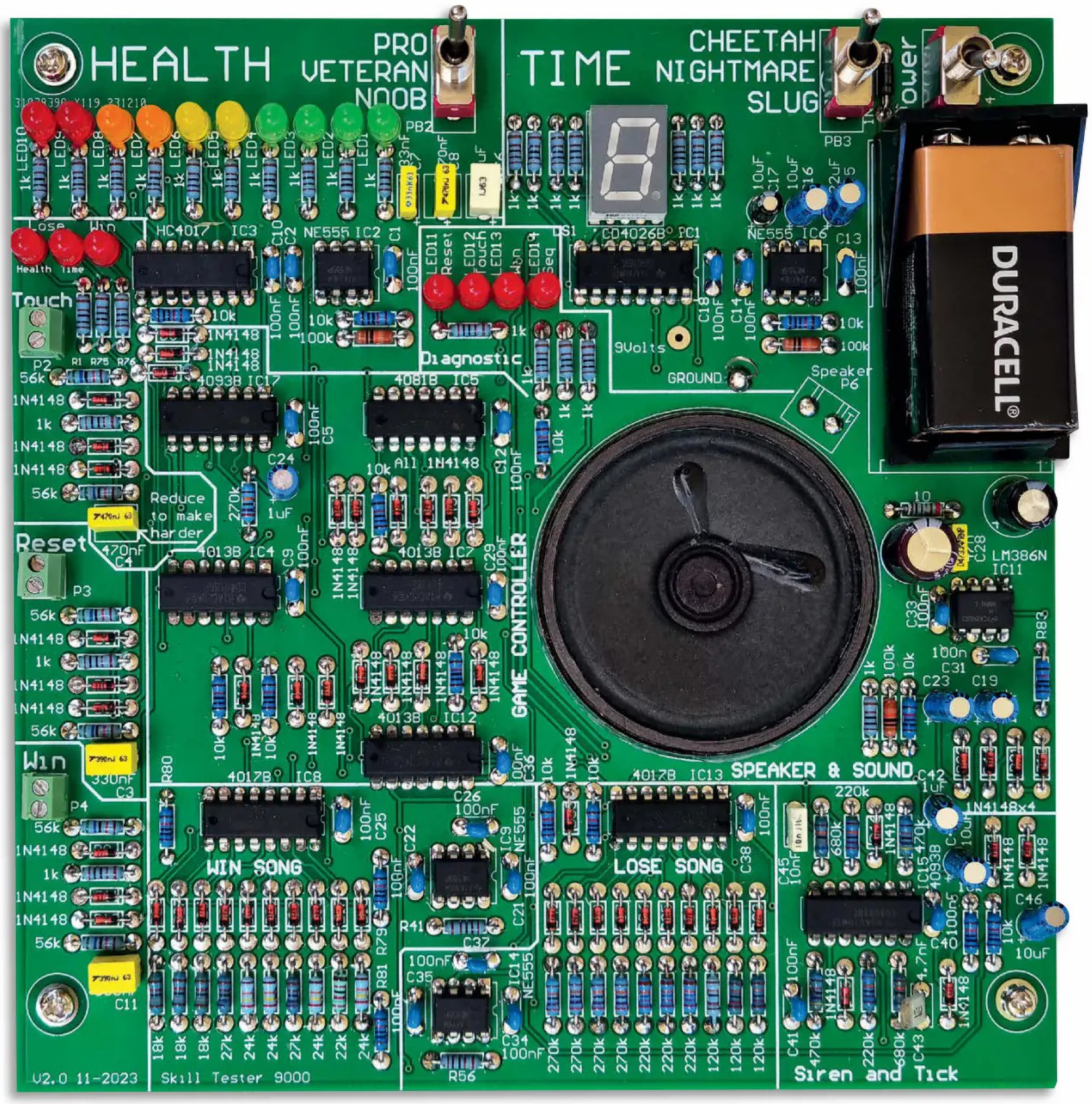
We use a NAND gate here because we have a Schmitt trigger input NAND gate IC, with positive-going and negative-going input voltage thresholds about 1V apart. The voltage difference or 'hysteresis' is needed for it to oscillate when we apply feedback.

To put it another way, let's say the voltage at the input is increasing from 0V, and at 5V, the output switches low. The input voltage then starts to decrease, but it has to drop to 4V before the output will go high again. The resistor and capacitor values and hysteresis voltage combine to determine the oscillation frequency.

So, with the 470kΩ resistor and 1µF capacitor, IC15a oscillates at about 1Hz. Its output produces a square wave that switches between 0V and V_{DD} (about 9V), which feeds pin 6 of IC15b. IC15b is also configured as an oscillator, and the time it takes to charge or discharge the 10nF capacitor to its threshold voltage depends on whether the output of IC15a is high or low.

In this way, IC15a causes IC15b to oscillate at alternating frequencies (like a siren).

The output of IC15b is gated by AND gate IC5d, controlled by the outputs of the three state latches and the touch



The completed PCB of the Skill Tester 9000. We recommend you assemble the PCB in sections as shown on the silkscreen.

buffer. When the player touches the loop on the wire, the Touch line goes high, allowing the signal from oscillator IC15b to pass through to pin 11 of IC5d and the Touch Sound Out line.

However, we only want touching the wire to produce a sound if the game has yet to be won or lost. Thus, if the game is in the Win state or one of the Lose states, the Touch line cannot pull pin 13 of IC5d high via the resistor because the Win or Lose latch is holding pin 13 of IC5d low via the associated diode.

The resistor and diodes create a crude but effective four-input AND gate (Touch AND $\overline{\text{Win Latch}}$ AND $\overline{\text{Time Lose}}$ AND $\overline{\text{Health Lose}}$).

The ticking sound

The Skill Tester 9000 makes a ticking noise every time the timer value decreases, from the initial value of zero until it reaches the terminal count of nine, and the game is lost. The circuit to generate the ticking noise is similar to that for Touch. While the Touch noise is supposed to be ‘angry’, we want the tick to create a sense of urgency and doom, but with just a little hope of finishing!

The siren oscillator circuit, based around IC15c and IC15d, is the same but set for higher frequencies. We have added an amplitude modulator based on the components between output pin 10 of IC5c and the Tick Out line, which softens the sound somewhat. In a sense, this is a poor person’s voltage-controlled attenuator, so very much in the spirit of this project!

It works as follows. Each time the Trigger line goes high, the two capacitors form a capacitive voltage divider, bringing the cathode of D47 to about half the supply rail voltage. This voltage decays as the capacitor discharges via the 10kΩ resistor to GND, or both 10kΩ resistors if pin 11 of IC15d is low. The result is a fast rise time with a slow, exponential falloff.

In this way, the ‘tick’ pulses on the Time CLK line amplitude modulate the ~20Hz waveform from the IC15c/d oscillator.

Winning and losing songs

How do we make a tune using 4000-series logic? Some may say that “tune” is generous. Others might think this is pretty cool. I find it amazing that parts like the 4000-series logic chips and 555 timers are half a century old and still in use.

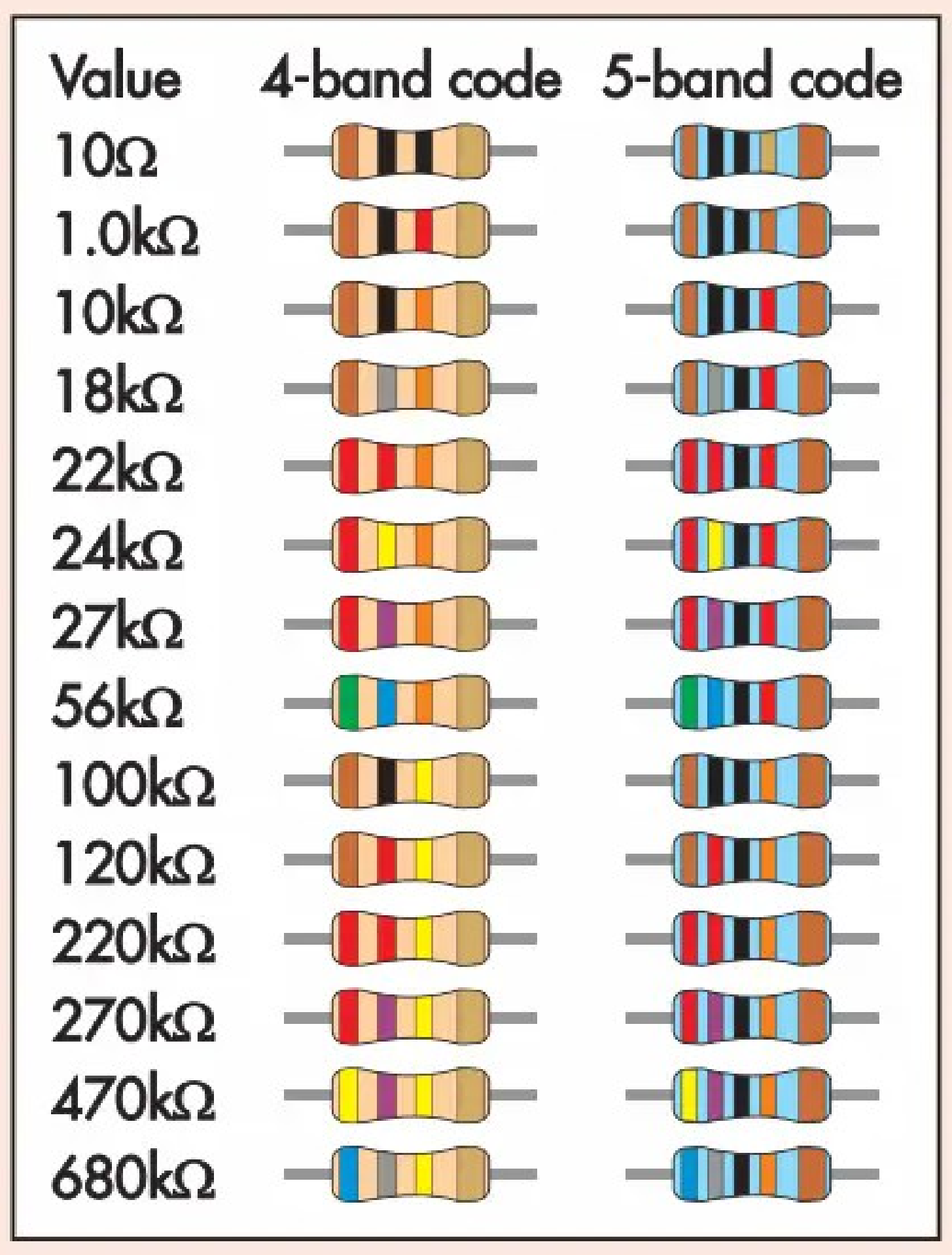
Parts List – Skill Tester 9000

1 double-sided PCB coded 08101241, 174 × 177mm
1 0.5in (12.7mm) common-cathode 7-segment LED display (DS1)
  [eg, Farnell 1142435 (red), 3779163 (super red) or 4228453 (blue)]
1 PCB-mounting vertical SPDT regular (on-on) toggle switch (S1)
2 PCB-mounting vertical SPDT centre-off (on-off-on) toggle switches (S2, S3)
1 PCB-mounting 9V battery holder (CON1) **[eg, Farnell 3126589]**
1 9V battery (alkaline recommended)
1 57mm 8Ω loudspeaker **[eg, Farnell 4411387]**
3 2-way mini terminal blocks, 3.5mm pitch (CON2-CON4) **[eg, Farnell 1708074]**
1 2-way mini terminal block, 5/5.08mm pitch (CON6)
4 16-pin DIL IC sockets (**optional**)
6 14-pin DIL IC sockets (**optional**)
5 8-pin DIL IC sockets (**optional**)

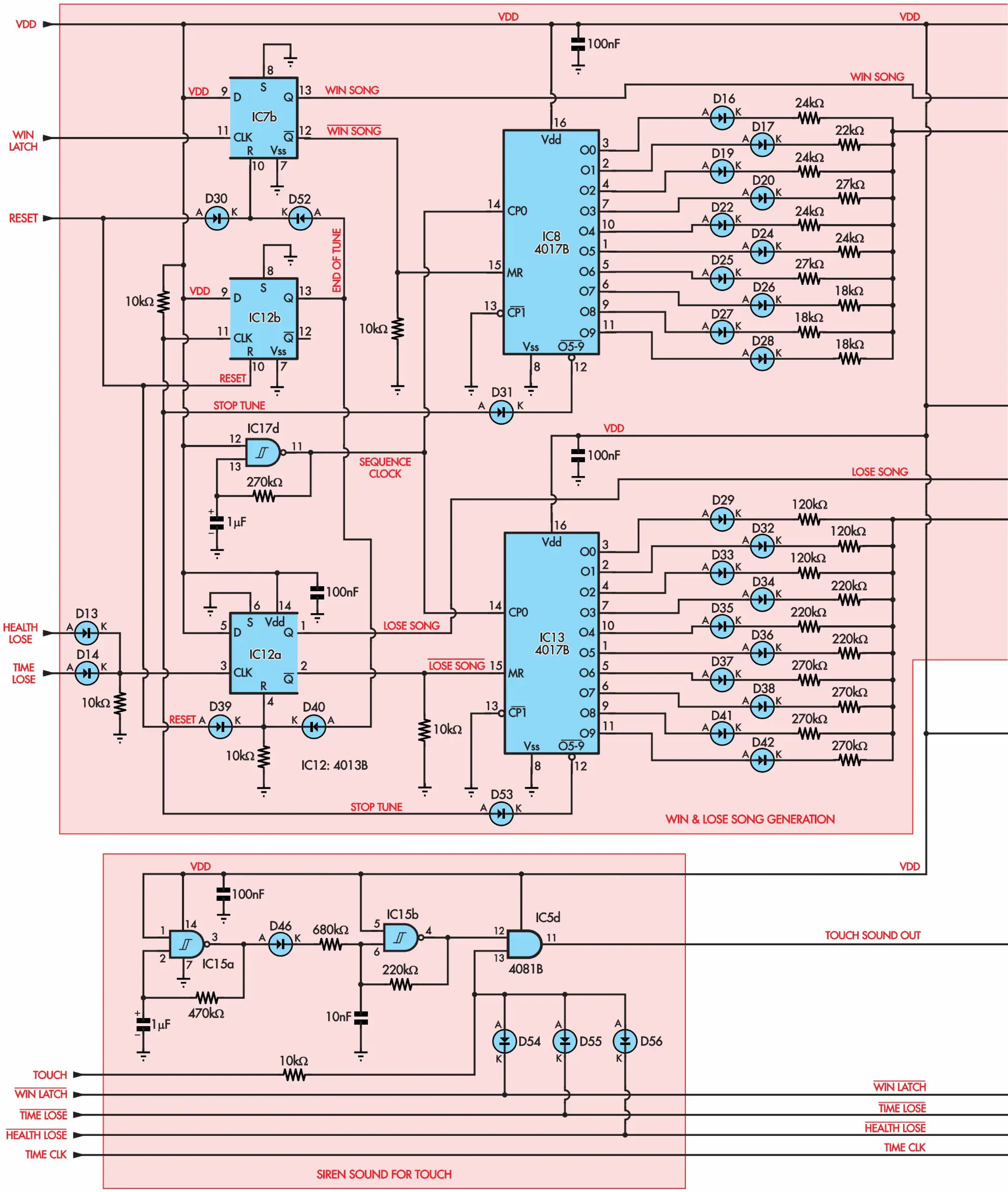
Hardware, wire etc
1 500 × 200mm × 12mm (approximately) timber baseplate
4 M3 × 16-25mm panhead machine screws (depending on baseplate thickness)
4 M3 × 6mm panhead machine screws
8 M3 shakeproof washers
8 M3 × 20mm tapped spacers
4 5mm or $^3/_{16}$in × 30mm gutter bolts
8 5mm or $^3/_{16}$in hex nuts
8 5mm or $^3/_{16}$in flat washers
1 1m length of 2mm diameter steel wire (eg, from a coathanger)
4 1m lengths of heavy-duty hookup wire (eg, red, black, blue & yellow)
1 1m length of super-flexible silicone-insulated cable (for the wand)
1 1m length of 1mm diameter tinned copper wire
4 50mm lengths of 4mm diameter heatshrink tubing
4 ring or fork crimp lugs (to connect wires to the board)
4 stick-on rubber feet
1 small tube of superglue

Semiconductors
1 4026B CMOS decade counter/divider, DIP-16 (IC1)
4 555 timers, DIP-8 (IC2, IC6, IC9, IC14)
3 4017B decade counter/divider, DIP-16 (IC3, IC8, IC13)
3 4013B dual D-type flip-flops, DIP-14 (IC4, IC7, IC12)
1 4081B quad 2-input AND gate, DIP-14 (IC5)
1 LM386N 1.25W mono audio power amplifier, DIP-8 (IC11)
2 4093B quad 2-input Schmitt-trigger NAND gates, DIP-14 (IC15, IC17)
4 green 5mm LEDs (LED1-LED4)
2 yellow 5mm LEDs (LED5, LED6)
2 amber/orange 5mm LEDs (LED7, LED8)
9 red 5mm LEDs (LED9-LED17)
55 1N4148 or 1N914 diodes (D1-D50, D52-D56)
1 1N5819 40V 1A schottky diode (D51)

Capacitors
1 470µF 16V radial electrolytic
1 220µF 16V radial electrolytic
1 22µF 50V radial electrolytic
6 10µF 50V radial electrolytic
2 1µF 50V radial electrolytic
1 1µF 63V MKT
2 470nF 63V MKT
2 330nF 63V MKT
23 100nF 50V multi-layer ceramic
1 47nF 63V MKT
1 33nF 63V MKT
1 10nF 63V MKT
1 4.7nF 63V MKT


Value
4-band code
5-band code
10Ω
1.0kΩ
10kΩ
18kΩ
22kΩ
24kΩ
27kΩ
56kΩ
100kΩ
120kΩ
220kΩ
270kΩ
470kΩ
680kΩ

Resistors *(all $^1/_4$W 1% unless noted)*
2 680kΩ
2 470kΩ
5 270kΩ
5 220kΩ
3 120kΩ
3 100kΩ
6 56kΩ
2 27kΩ
4 24kΩ
1 22kΩ
3 18kΩ
16 10kΩ
30 1kΩ
1 10Ω


VDD
WIN LATCH
RESET
HEALTH LOSE
TIME LOSE
IC7b
IC12b
IC12a
IC12: 4013B
IC17d
270kΩ
1µF
100nF
10kΩ
D30
D52
D13
D14
D39
D40
D31
D53
END OF TUNE
STOP TUNE
SEQUENCE CLOCK
WIN SONG
LOSE SONG
IC8 4017B
IC13 4017B
CP0
MR
CP1
Vdd
Vss
O0
O1
O2
O3
O4
O5
O6
O7
O8
O9
O5-9
D16
D17
D19
D20
D22
D24
D25
D26
D27
D28
24kΩ
22kΩ
27kΩ
18kΩ
D29
D32
D33
D34
D35
D36
D37
D38
D41
D42
120kΩ
220kΩ
270kΩ
WIN & LOSE SONG GENERATION
IC15a
IC15b
IC5d
4081B
D46
680kΩ
220kΩ
470kΩ
10nF
D54
D55
D56
TOUCH SOUND OUT
TOUCH
WIN LATCH
TIME LOSE
HEALTH LOSE
TIME CLK
SIREN SOUND FOR TOUCH

SC ©2024 **SKILL/DEXTERITY GAME** SOUND GENERATION & AMPLIFIER SECTION

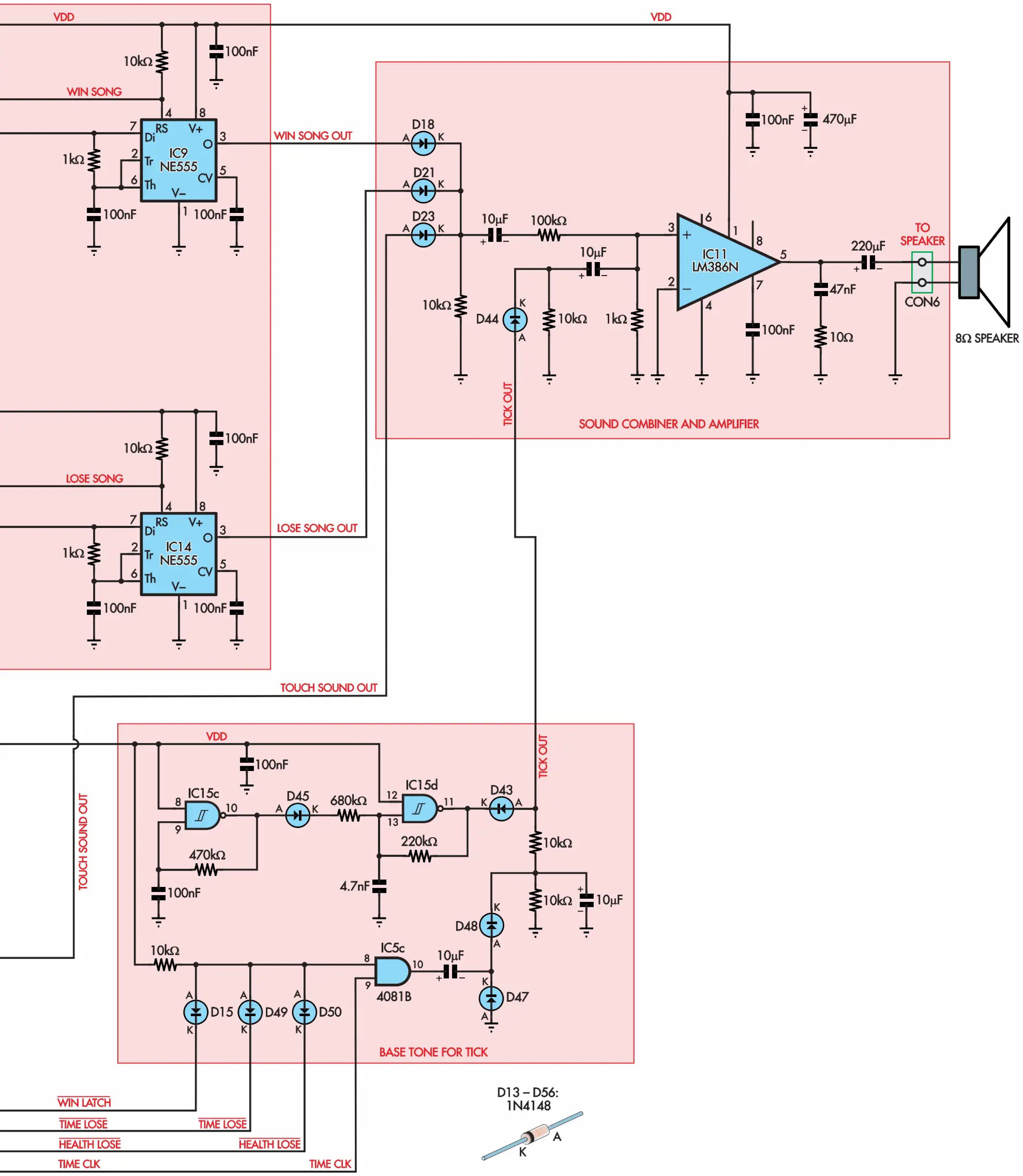

VDD
10kΩ
100nF
WIN SONG
IC9
NE555
RS
V+
Di
O
Tr
CV
Th
V–
1kΩ
100nF
100nF
WIN SONG OUT
VDD
D18
D21
D23
A
K
10µF
100kΩ
10µF
10kΩ
D44
10kΩ
1kΩ
TICK OUT
IC11
LM386N
100nF
470µF
220µF
47nF
100nF
10Ω
TO SPEAKER
CON6
8Ω SPEAKER
SOUND COMBINER AND AMPLIFIER
10kΩ
100nF
LOSE SONG
IC14
NE555
1kΩ
100nF
100nF
LOSE SONG OUT
TOUCH SOUND OUT
TOUCH SOUND OUT
VDD
100nF
IC15c
D45
680kΩ
IC15d
D43
TICK OUT
470kΩ
220kΩ
10kΩ
100nF
4.7nF
10kΩ
10µF
D48
10kΩ
IC5c
4081B
10µF
D47
D15
D49
D50
BASE TONE FOR TICK
WIN LATCH
TIME LOSE
TIME LOSE
HEALTH LOSE
HEALTH LOSE
TIME CLK
TIME CLK
D13 – D56:
1N4148
A
K

Fig.3: the remainder of the circuit is dedicated to producing the various sounds. The Win song generator (top left) and Lose song generator (below) are similar but use different resistor values to produce different tunes. The touch siren and time tick sections are shown below those, and the output of the four sound generators are mixed using diodes, feeding power amplifier IC11 to drive the speaker.

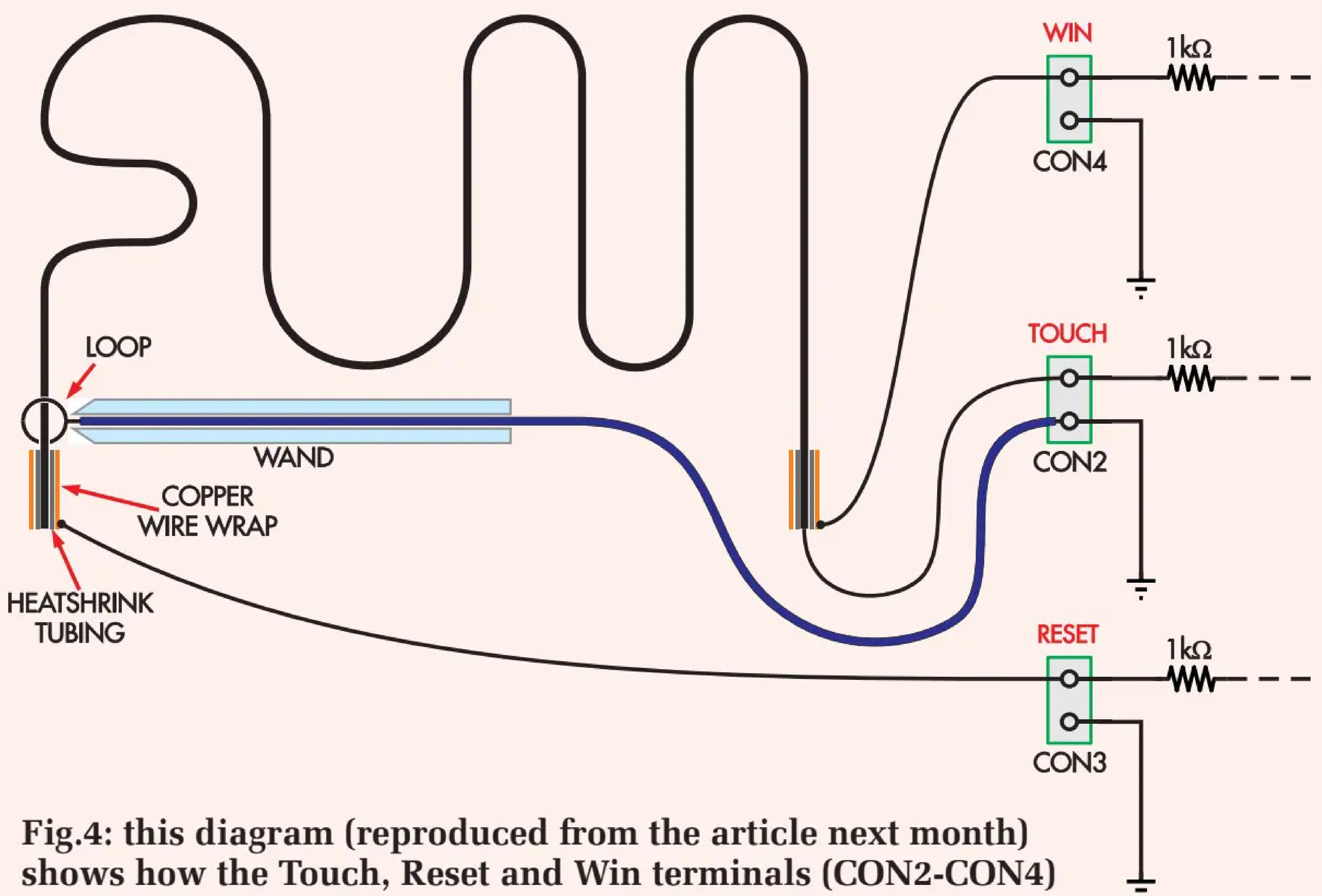


Fig.4: this diagram (reproduced from the article next month) shows how the Touch, Reset and Win terminals (CON2-CON4) connect to the wand, game wire and start and finish pads. Note that the ground wire going to the wand can connect to the lower screw of any of the three terminals.

Fundamentally, the tunes are generated by a 555 timer set up in an astable oscillator configuration. That's a fancy way of saying "free running". The frequency of operation is defined as $f = 1 \div (C \times [R_a + 2 \times R_b])$.

In our circuit, $R_a = 1k\Omega$, $C = 100nF$ and R_b is the resistance in series with the diodes from the 4017B (IC8 after winning or IC13 after losing). As the 4017B IC counts from 0 to 9, only one of its Q output pins is high at a time. The high output becomes the charging source for the 100nF capacitor in the 555 timer circuit, and the series diodes stop the other resistors from loading this down.

This means we can set 10 different frequencies that the 555 oscillates at in sequence to make notes in our tune. The clocks for the 4017 that set the note pace/duration come from the sequence clock, another simple Schmitt-trigger oscillator based on IC17d.

The 4017 ICs will count from 0 to 9 and then back to 0, repeating forever if we don't stop them. To stop the tune after the 10th note, we use an extra flip-flop per 4017 IC, triggered by the 4017's carry output. When triggered, the flip-flop latches the reset input of the 4017 and the $\overline{\text{RESET}}$ input of the corresponding 555 oscillator (they are different).

After being reset, the $\overline{Q}$ output of the flip-flop (IC7b or IC12a) is high and the Q output is low. This holds both the 4017 and 555 in reset, so they are initialised but doing nothing. When the Win Latch or one of the Lose latches goes high, that clocks the WinSong or LoseSong flip-flop, taking the 4017 and 555 out of reset, and they start playing the 10 notes.

The 4017's carry output (CO) goes low after five notes and goes high again after 10. By combining these carry-out signals through two diodes, which are pulled high by 10kΩ resistors, we can use the carry-out lines from both 4017 counters to trigger End Of Tune as it is that final rising edge that the D-type flip-flop uses.

This End Of Tune signal resets the WinSong and LoseSong flip-flops, putting both the 4017 and 555 ICs back into reset, thus stopping the tune.

If you analyse the circuit, you will see two 10kΩ resistors that do nothing in regular operation, at the reset and $\overline{\text{RESET}}$ inputs of the 4017 and 555, respectively. We have included these so we can test the circuit before all parts have been mounted on the board; the final controller chips are added at the last stage.

The timer

To limit the game time, we are using a 4026 decade counter that can drive a 7-segment LED display. To minimise the parts count, we simply use unbuffered series resistors for the LEDs, which achieves good brightness but will cause reduced output voltage from the 4026 due to loading the CMOS outputs.

The 4026 IC needs a clock, which we generate using a 555. This allows us to switch in different timing capacitors to make slow, medium and fast difficulty levels. The clock rate is about 1.5Hz on the quickest setting, giving a total of six seconds. On the slowest settings, each count is a little under 4 seconds, for a total of around 30 seconds.

If you wish to change these speeds, you can change the values of the 10μF and 22μF capacitors.

The Timer clock is cleared by the system reset line, ensuring that at the start of each game, the timer starts at 0. The Clock Enable input is driven by our combined Win Lose Latch signal that goes high if any of the Win, Time Lose or Health Lose latches goes high. This way, if the game ends for any reason, this timer stops.

The only output from this circuit is the carry-out signal from the 4026, which in our circuit is labelled Out Of Time. This drives the clock input to the Time Lose latch; the rising edge of this ends the game.

Health

The 'health' status in games is usually a bargraph that goes from green to red. We use the now-familiar 4017 decade counter IC for this (IC3). In this case, we have connected LEDs to its output rather than clocking it to make a tune. That allows us to get creative with the LED colours.

The 4017 is not intended to drive LEDs, but it does OK, provided you don't want the 4017 outputs to drive other CMOS logic reliably, as the voltages will droop.

The health counter is implemented like a hit counter. The longer you touch the wire, the more hits you take. We have implemented this by using the Touch input to enable a 555 timer. With that input low, the 555 (IC2) is in reset and produces no output pulses. While the Touch line is high, the 555 oscillator runs free.

The output of the 555 drives the clock input to the 4017 counter. After either 10 touches or a time period long enough for 10 counts of the 555, the hit counter reaches zero health and Carry Out goes high. The remainder of the logic around this is identical to the Time counter.

In this case, we have set the clock rate for the 555 to a much faster pace. With the same 10kΩ and 100kΩ resistors for R_b and R_a, the slow count rate is set by a 1μF capacitor in parallel with a 33nF capacitor, resulting in 6.6Hz, allowing about 1.5 seconds of touch. The fast count runs at about 150Hz, so pretty much any touch ends the game.

You can change these capacitor values. If you want to use electrolytics in these locations, you can; we have marked the "+" end of each on the silkscreen.

The selection of LED colours warranted some discussion with my helper. The advice is that it definitely starts with green and ends with red. In between are as many colours as you can get a hold of. We have recommended using red, amber, yellow and green in the parts list. You can tweak the series resistor values if some are too bright or dim.

Audio output

The audio output section has a mixer/combiner implemented using more 1N4148 diodes. The output is pulled to ground with a 10kΩ resistor, then capacitively coupled to the LM386 amplifier. Its gain has been set to produce a generous sound level. Note that this diode mixer only works because we are combining digital signals. This circuit takes lo-fi to new levels!

If you want to reduce the volume, we suggest adding a series resistor for the loudspeaker. 100Ω 1W would be a good place to start.

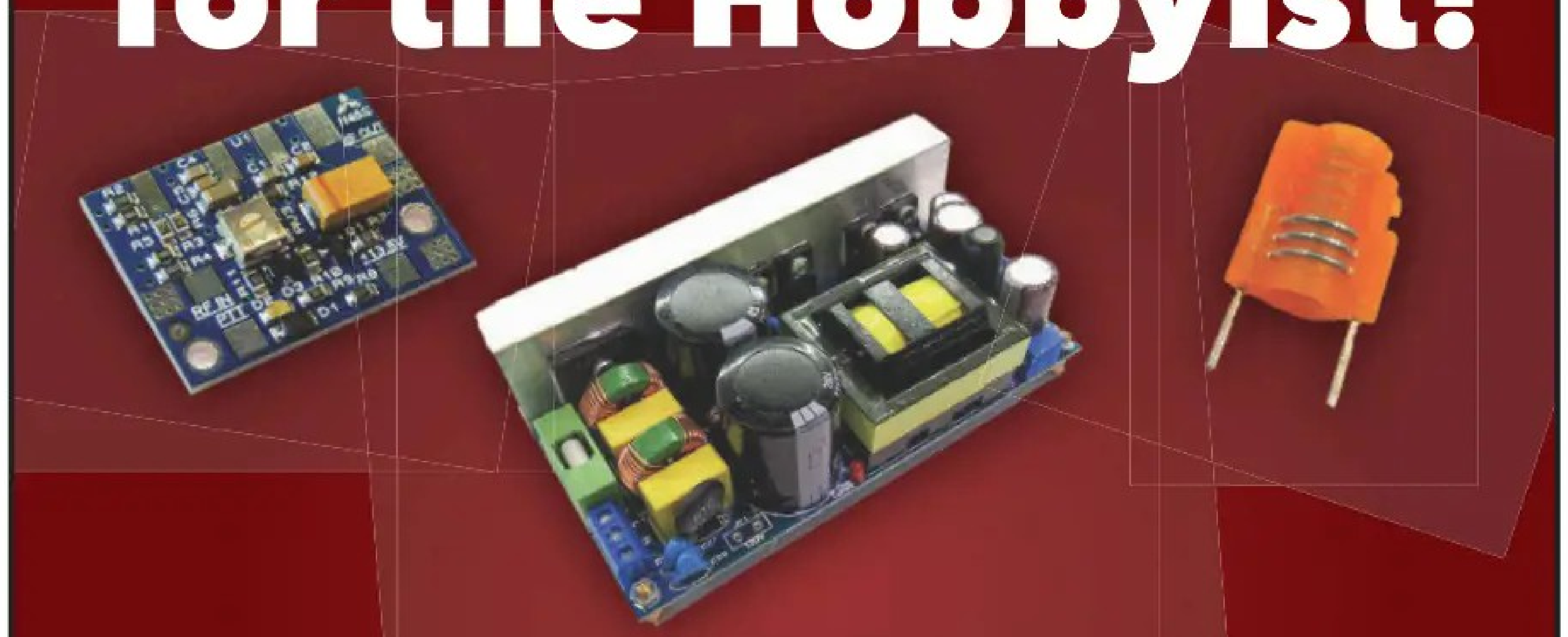
To keep assembly simple, we have put a cutout on the PCB that will accept a 57mm speaker, which can be glued in place with super glue, Araldite or whatever comes to hand.

Input debouncing

Earlier on, we skipped over some of the details of how we detect touches on different parts of the wire in favour of explaining the game logic. The Win, Touch and Reset inputs have identical debounce circuits. When a switch closes, it is never perfect, and the connection 'bounces' for a few milliseconds. Many digital circuits are so fast that such bouncing can interfere with their operation.

In each case, our input starts with a 1kΩ series resistor and normally reverse-biased diodes to ground and V_{DD}. This protects the circuit from static, which we expect will be present with enthusiastic hands and feet on the carpet. The inputs have a 56kΩ pull-up resistor and a 470nF capacitor to GND, which gives a time constant of 26ms. The arrangement of two 56kΩ resistors makes it roughly the same for rise and fall.

This signal feeds a Schmitt-trigger input buffer, adding further immunity to bounce through its ~1V input hysteresis. The output of the Schmitt triggers goes to the game control logic and debugging LEDs, which let you see that these inputs are working.

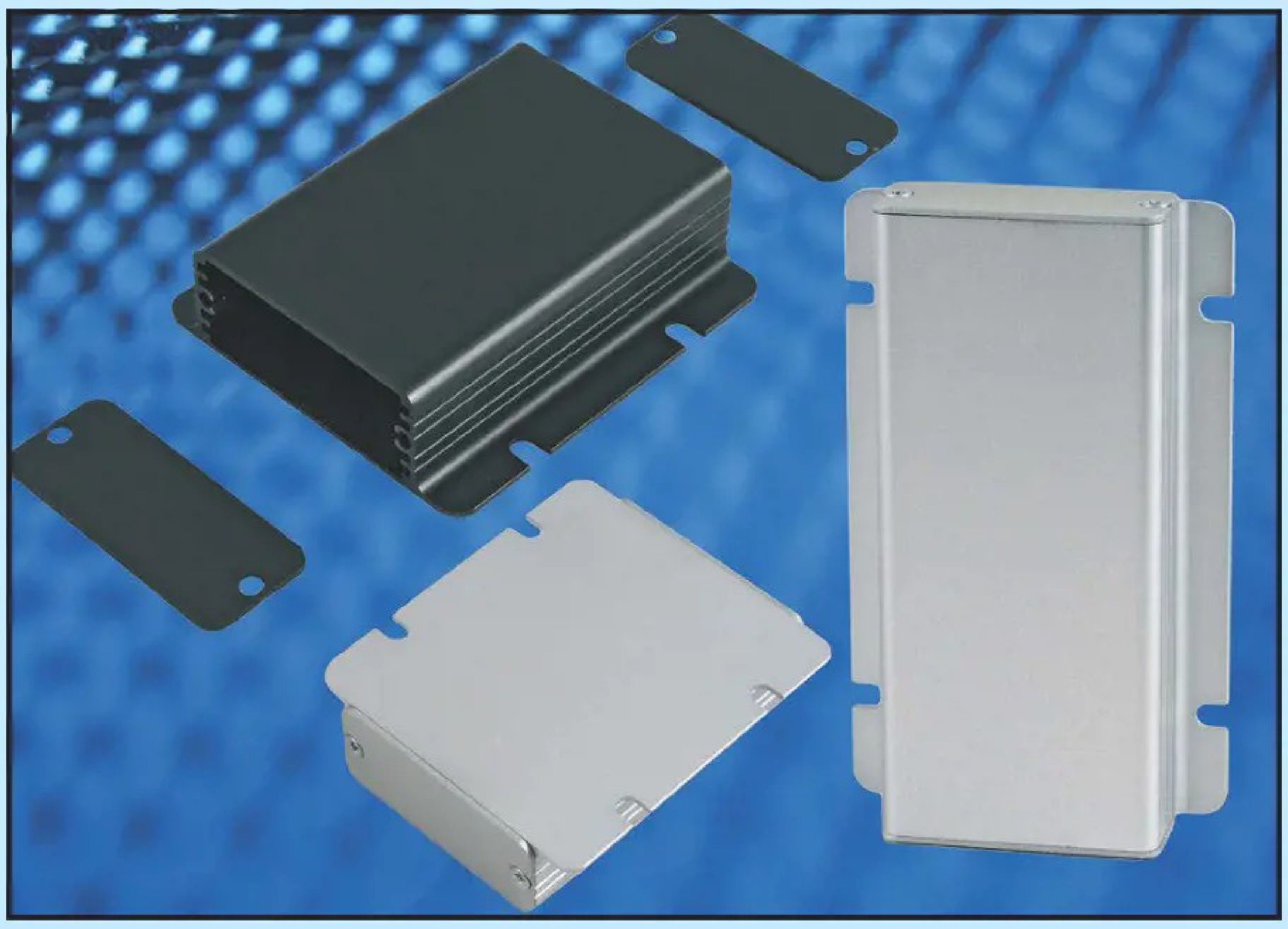


ENIGMA
COMPONENT SHOP
www.enigma-shop.com
Electronic Components for the Hobbyist!
Cable & Wire • Capacitors • Connectors • Filters
Kits & Modules • Hardware & Cooling
and much more...
READY TO BE REWARDED?
Join our loyalty scheme and **earn points** on every purchase you make which can be **used for discounts** in the future.
Read our reviews on ★ Trustpilot

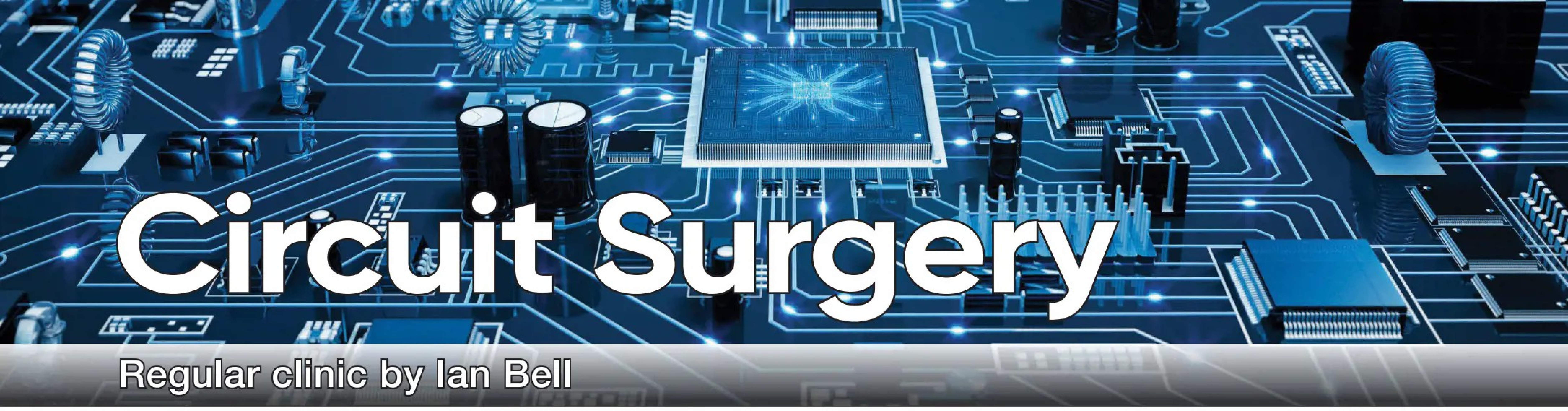
Next month

Next month's second and final article in this series will give all the construction details, including the PCB overlay diagram and how to make the wire and attach everything to the baseplate.

Importantly, PCB construction is broken up into stages, and you can test new functions at the end of each stage. We'll also have some hints on troubleshooting and how to play the game, including tournament rules. PE



HAMMOND®
1455F extruded flanged enclosures
Learn more:
hammondmfg.com/1455f
uksales@hammfg.com • 01256 812812



Circuit Surgery

Regular clinic by Ian Bell

Topics in digital signal processing – high-pass, band-pass and band-stop filters

We are looking at various topics related to digital signal processing (DSP).

DSP covers a wide range of electronics applications where signals are manipulated, analysed, generated, stored or displayed as digital data but originate from and/or are converted to real-world signals for interaction with humans or other parts of the physical world.

Fig.1 shows the key elements of a generic DSP system with a signal path from an analog input via digital processing to an analog output. This does not necessarily represent every DSP system (not all have all the parts shown), but it serves as a reference for the various subsystems we will consider.

Over the last few months, we have explored various aspects of digital filters (the processing part in Fig.1), concentrating on discussing windowed sinc filters, as these are relatively straightforward to implement and understand.

As well as covering the theory behind their operation, we have shown how the filter design calculations (finding their coefficient values) can be performed using spreadsheets and how LTspice can be used to simulate filter operation.

Last month, we discussed creating symbols for the filters in LTspice so that the subcircuit code used to implement them could be easily reused in different circuit simulations.

Beyond low-pass responses

We started the discussion on filters with the simple case of a moving average, which is a well-known data smoothing operation and, in circuit terms, provides a low-pass function. The moving average is a weighted sum of the current and past inputs, which leads to the structure shown in Fig.2.

This is known as a finite impulse response (FIR) filter because an impulse (input signal comprising just one sample at value one) produces an output that is non-zero for a finite duration.

A moving average uses equal weights (coefficients, a_n in Fig.2), but better filters can be obtained by using different values for individual coefficients. The coefficients for an ideal low-pass (brick wall) filter can be obtained from the inverse Fourier transform of its frequency response, which is found to be the sinc function.

A real implementation has a limited number of coefficients (N), but simply truncating the set of values does not give the best results. Instead, we apply a window function to smoothly reduce the values towards the ends of the used range.

So far, we have only looked at low-pass filters as they are the most straightforward in terms of theory and mathematics. However, high-pass, band-pass and band-stop (or band-reject) responses are also important in many systems. All of these can be implemented as windowed sinc FIR filters using the same structure as the low-pass (Fig.2).

This month, we will look at how to get these other filter response types.

From low-pass to high-pass

Using FIR filters structured as shown in Fig.2, we can achieve different response types. There is no need to change the fundamental structure; we just need to find the right set of coefficients (filter kernel) to obtain the desired response. The number of coefficients does not affect the response type – just the quality of the response (cut-off sharpness) and delay through the filter.

Usefully, if we have the coefficients for a low-pass response, we can get the kernel for a related high-pass filter from it with some minor additional steps (another reason we first considered low-pass filters). Furthermore, band-pass and band-stop filters can be obtained by combining high-pass and low-pass responses.

From a graphical perspective, high-pass filter responses can be obtained from related low-pass responses in a couple of ways. Firstly, we can flip the response curve upside down, as shown in Fig.3. This is referred to as spectral inversion. The high-pass and low-pass cutoff frequencies are the same ($f_{hp} = f_{lp}$).

Alternatively, we can reflect the response curve horizontally about a certain frequency. This is called spectral reversal and is illustrated in Fig.4, with the reflection line shown in red at half the Nyquist frequency (ie, half the sampling frequency [$f_s/2$] and the maximum frequency that we can process).

The relative cutoff frequencies depend on the reflection point, but in Fig.4 with the reflection around $f_s/4$, we get $f_{hp} = f_s/2 - f_{lp}$.

The preceding discussion defines how a low-pass response could be made into a high-pass response, but does not explain how it might be achieved in practice.

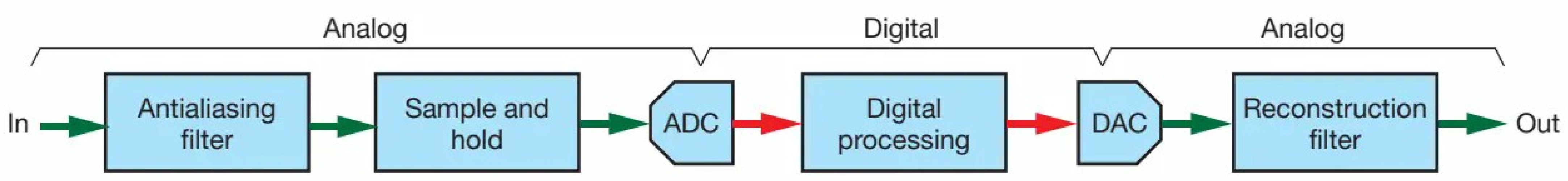


Fig.1: a generic digital signal processing (DSP) system structure.

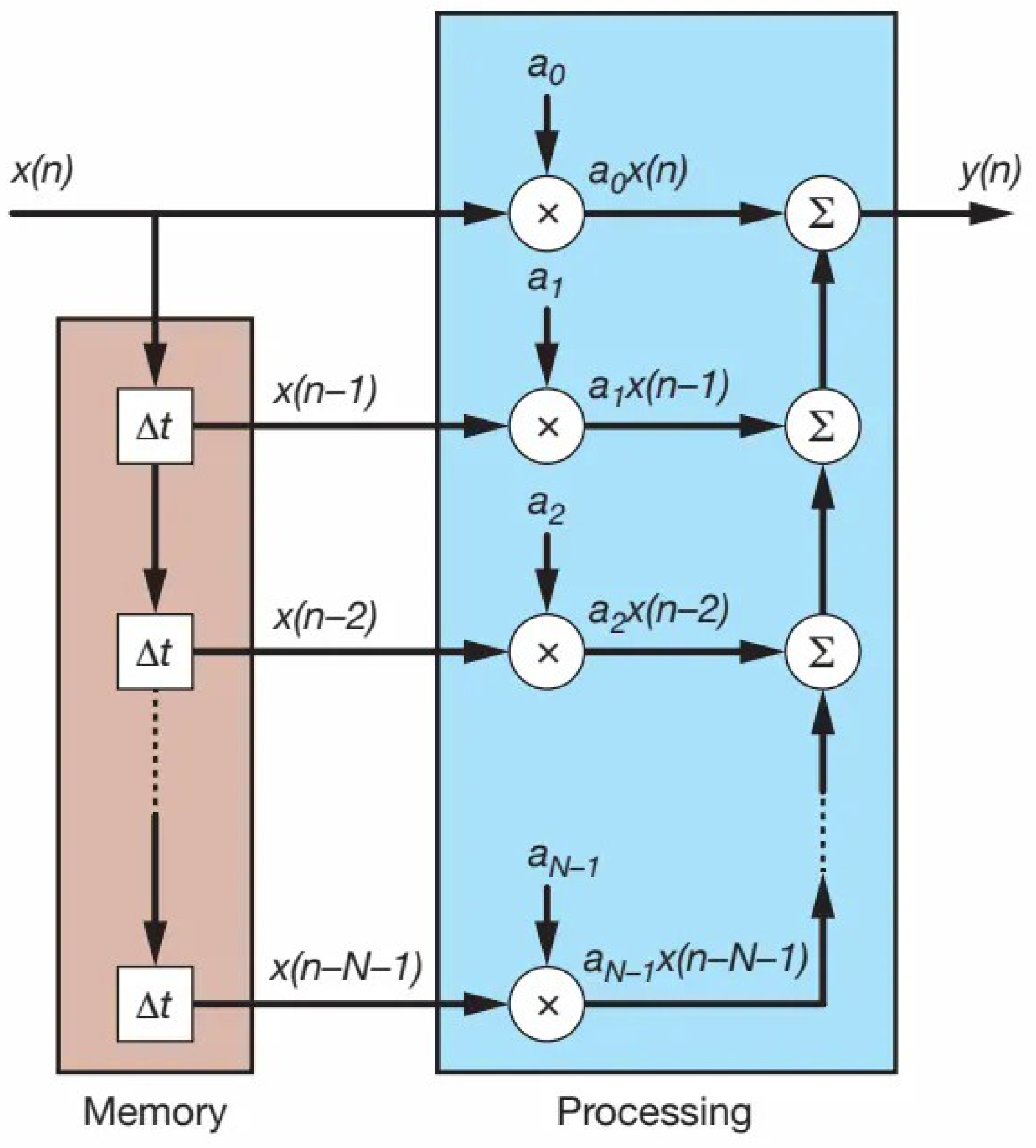


Fig.2: the structure of a finite impulse response (FIR) filter.

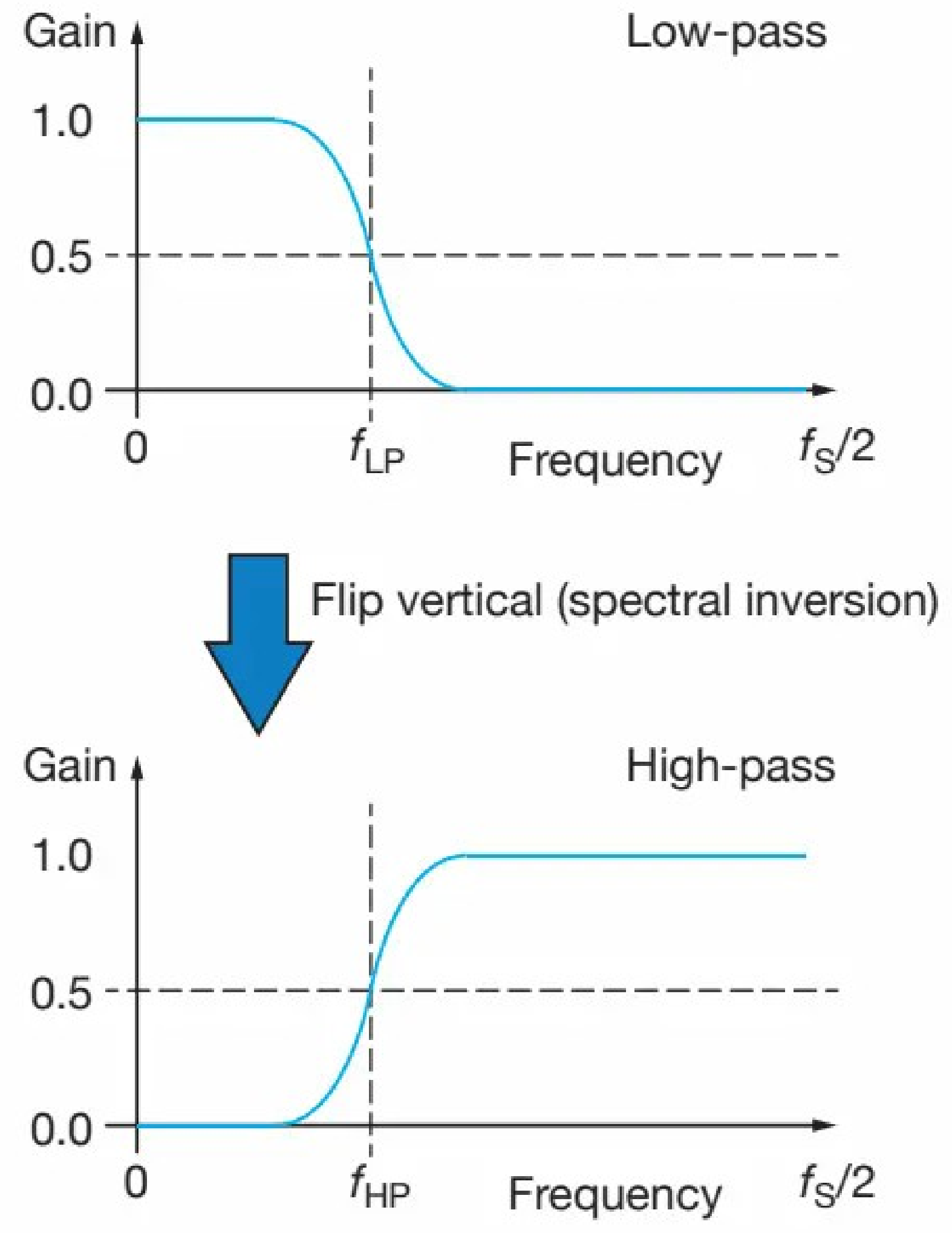


Fig.3: spectral inversion, a vertical flip of the low-pass frequency response, results in a high-pass response.

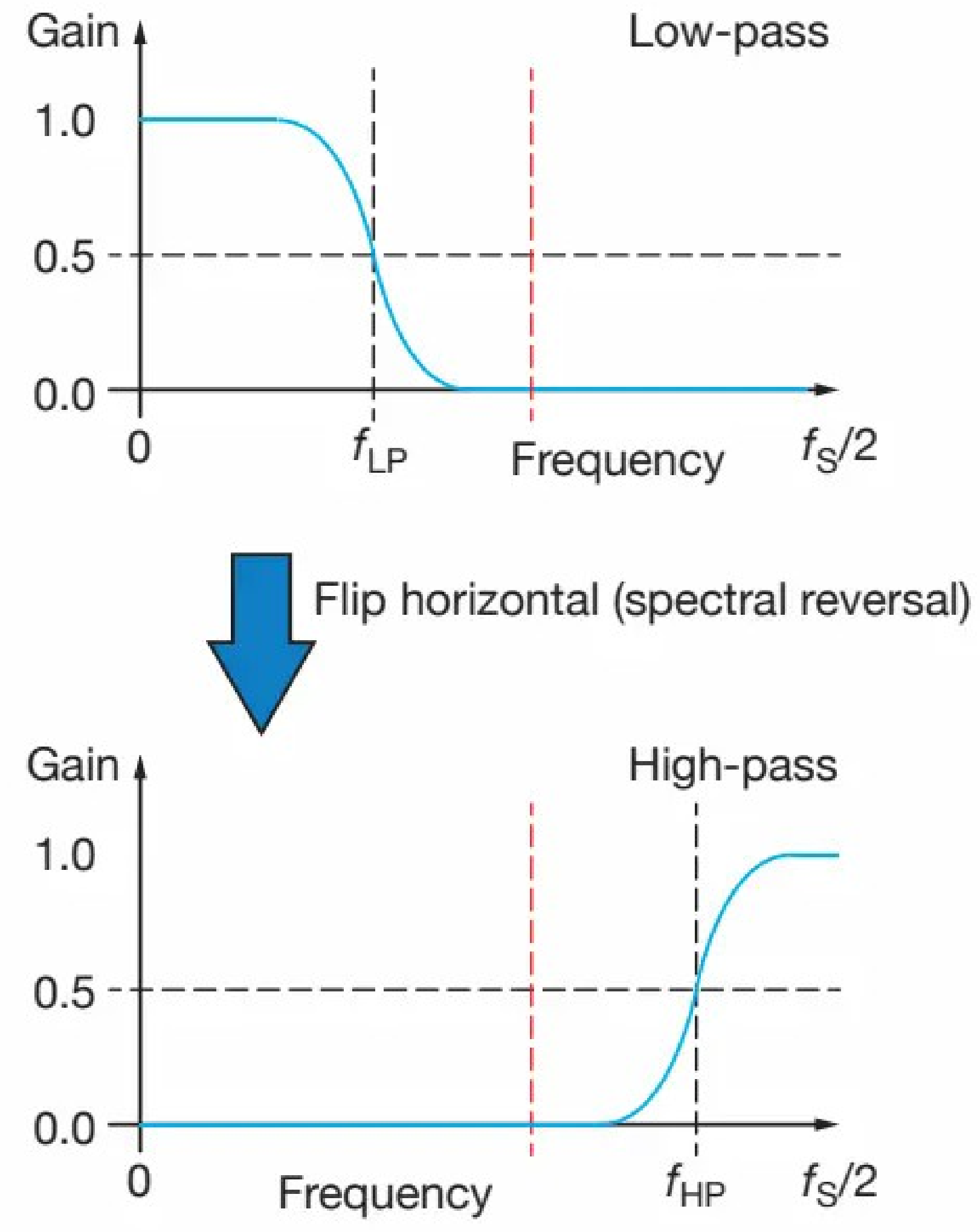


Fig.4: spectral reversal, a horizontal flip of the low-pass frequency response, also produces a high-pass response.

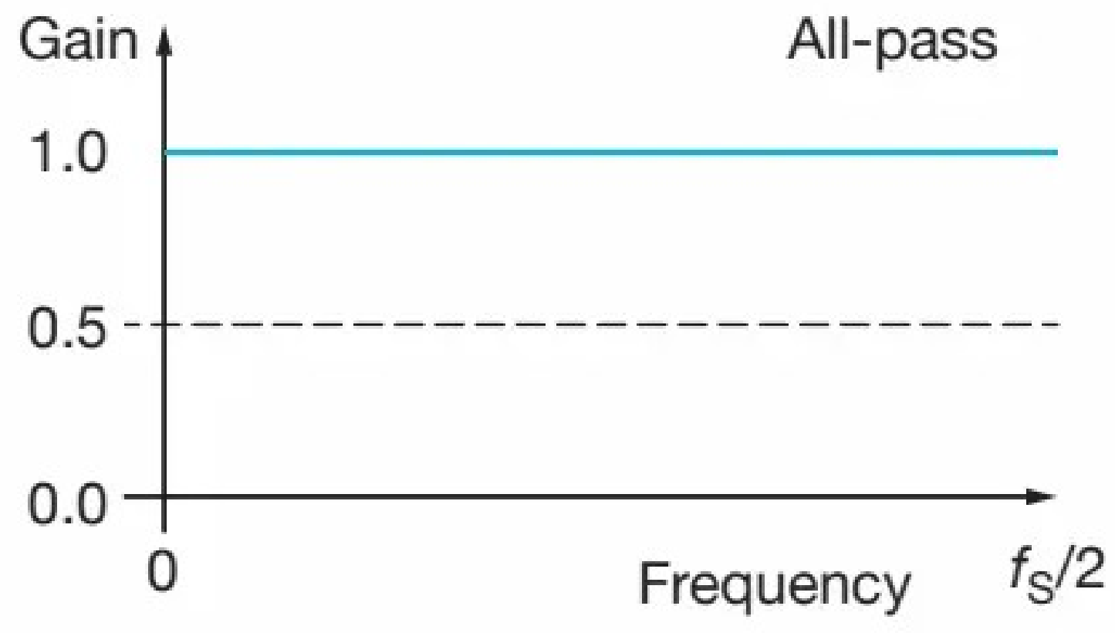


Fig.5: the all-pass filter response.

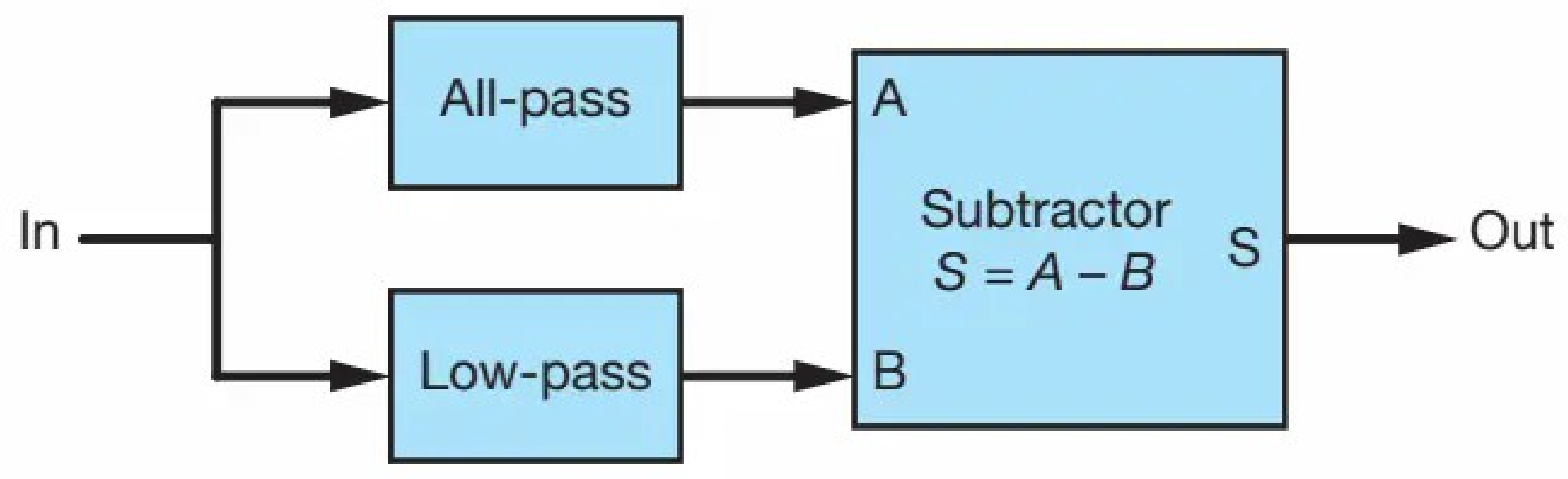


Fig.6: a possible high-pass filter implementation by spectral inversion.

We need to know how to calculate the high-pass coefficients from the low-pass coefficients in both cases.

High-pass by spectral inversion

Starting with spectral inversion, we note that this can be achieved by subtracting the output of a low-pass filter from a circuit with a flat frequency response of gain 1 (assuming the low-pass response varies from 0 to 1, as shown in Fig.3). The flat response circuit is called an all-pass filter (Fig.5).

Initially, it might seem that this could be just an ideal wired connection, but there is more to it than that, because we have to consider delay.

We can envisage implementing the high-pass filter using a circuit (or software processing) like that show in Fig.6. The signal is processed by the all-pass and low-pass functions in parallel, then a subtraction operation is performed to achieve the high-pass response.

How do we implement the all-pass function? Consider the structure in Fig.2. If we set a_n to one and all the other coefficients to zero, the output will be equal to the input with no variation in gain with frequency – a flat frequency response, as required.

In fact, we can set any of the coefficients to one with the others zero and achieve the same result. The difference is in the delay of the signal from input to output. The delay is equal to n sample periods if coefficient a_n is the non-zero value.

In order for the subtraction shown in Fig.6 to work correctly, the all-pass and low-pass blocks must have the same delay. Subtracting signals at different time points would lead to signal distortion.

As discussed in the February 2025 issue, the fundamental calculation of the sinc function coefficients for the low-pass filter gives a filter with zero delay but only if future values of the signal can be used. Such a non-causal filter cannot be implemented with real-time processing. The solution is to time shift the coefficients by $(N-1)/2$ samples for a filter with N coefficients.

The resulting filter has the same delay as the time shift. This means that to match the low-pass filter delay, the all-pass filter must have coefficient at index $(N-1)/2$ equal to one, and all others zero.

Fig.7 shows the raw coefficients for a 10kHz low-pass filter with 25 coefficients and a 48kHz sampling rate, similar to the previous examples we have looked at. Fig.8 shows the time shifted coefficients that form the causal filter kernel. The shift is $(25-1)/2 = 12$ samples, so the index range of kernel is $n = 0$ to 24.

Fig.9 shows the kernel required for the all-pass filter in Fig.6 to match its delay to the low-pass filter kernel in Fig.8.

Recall from discussions in previous months that the coefficients of a filter are equal to its impulse response. Looking at Fig.9, we see that the impulse response of the all-pass filter is an impulse function, so you may see this filter notated as $\delta[n]$.

The high-pass filter does not have to be implemented using the structure shown in Fig.6. This is useful for understanding the concept behind its operation, but it is more effective to perform the subtraction operation directly on the two kernels. This results in a single kernel that will provide the high-pass function without the need to use multi-step processing.

Doing this is straightforward – we just subtract the low-pass coefficients from the all-pass coefficients at each index. For all the coefficients, apart from the middle one, the all-pass values are zero, so we obtain the high-pass kernel simply by changing the sign of the low-pass coefficients.

For the middle one (where the all-pass coefficient is equal to 1), we also change the sign of low-pass coefficient, but

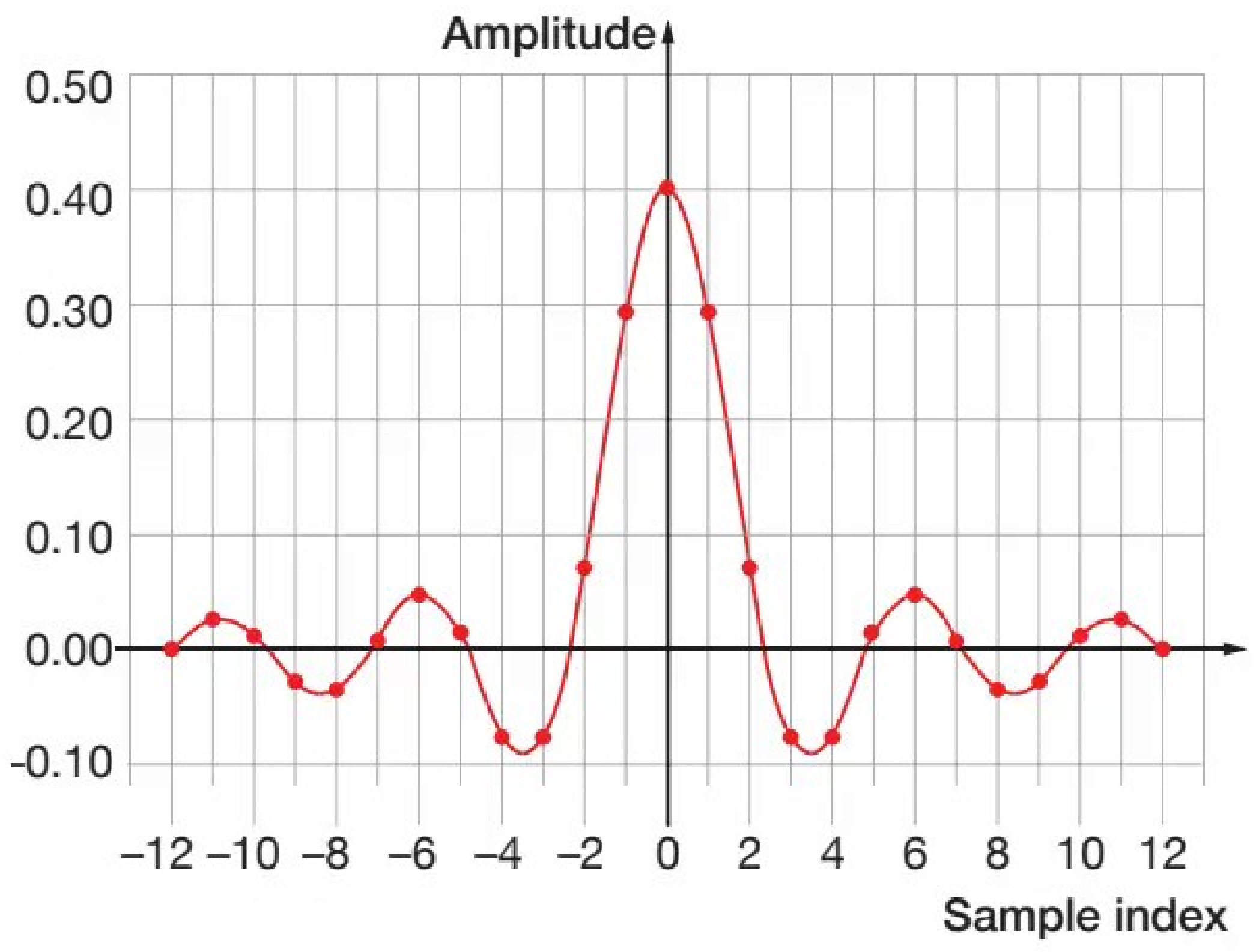


Fig.7: the raw coefficients for an example low-pass filter.

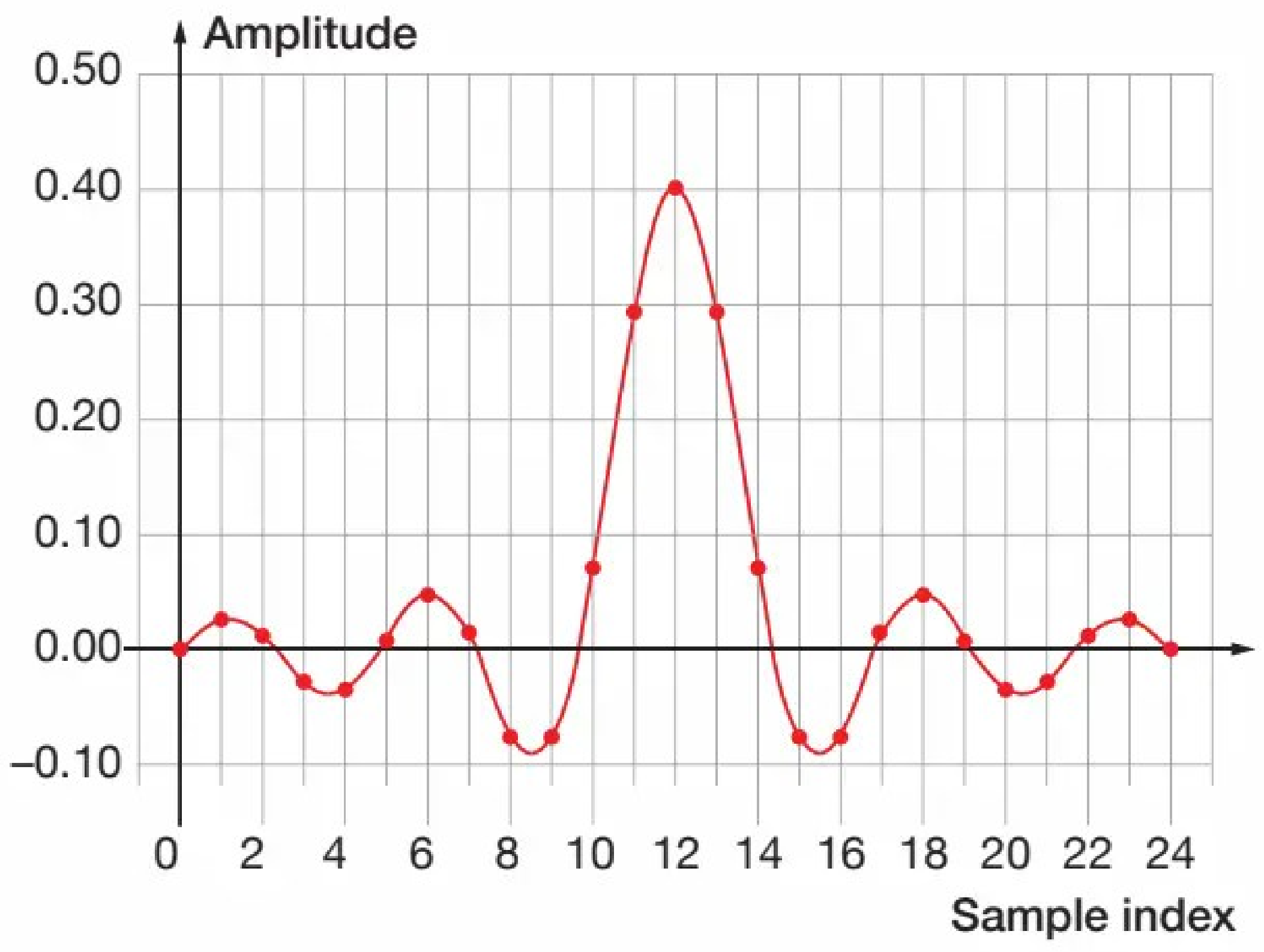


Fig.8: the coefficients from Fig.7 shifted to give a kernel for a causal filter.

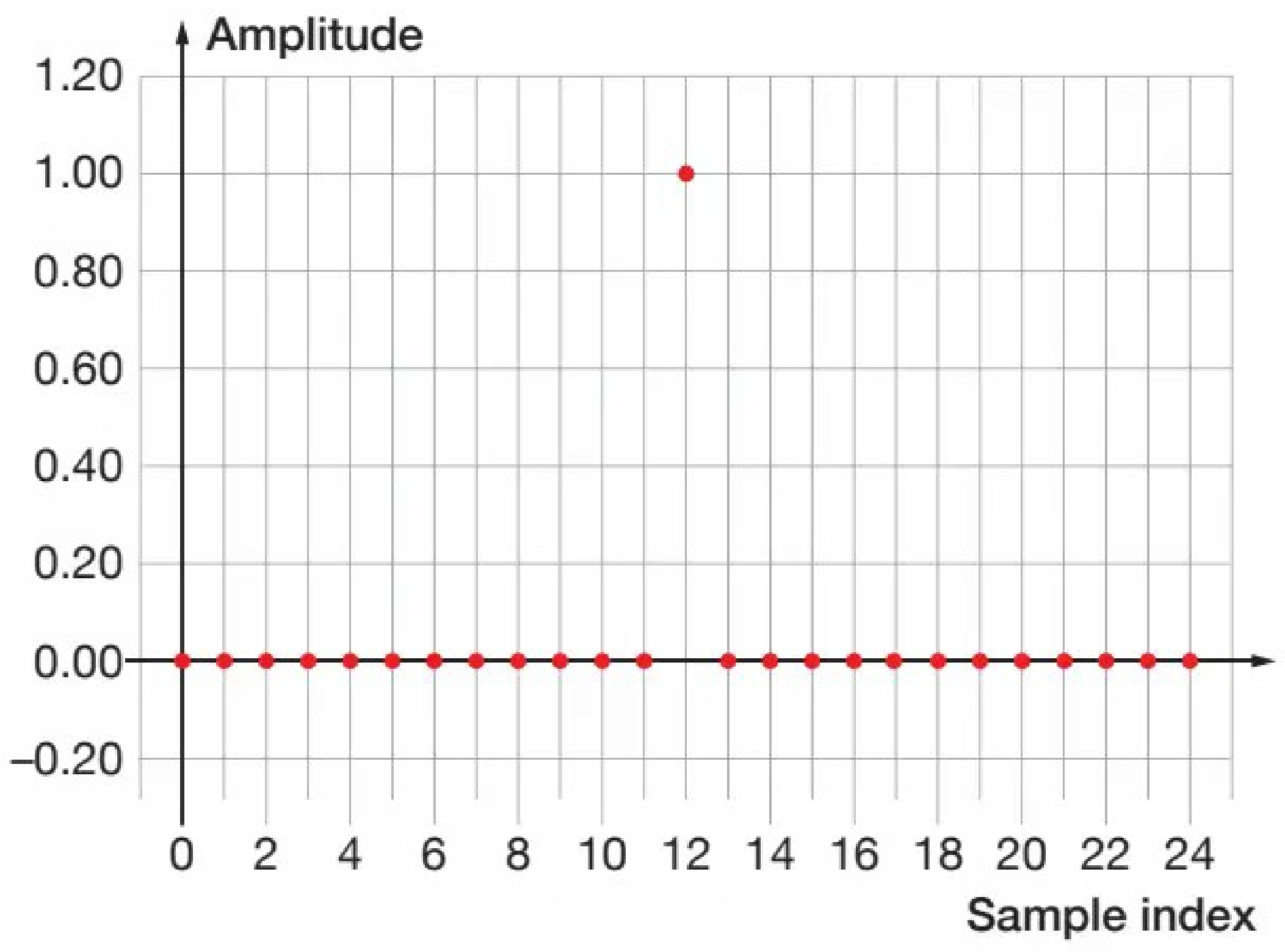


Fig.9: the kernel for the all-pass filter to match the low-pass filter in Fig.8.

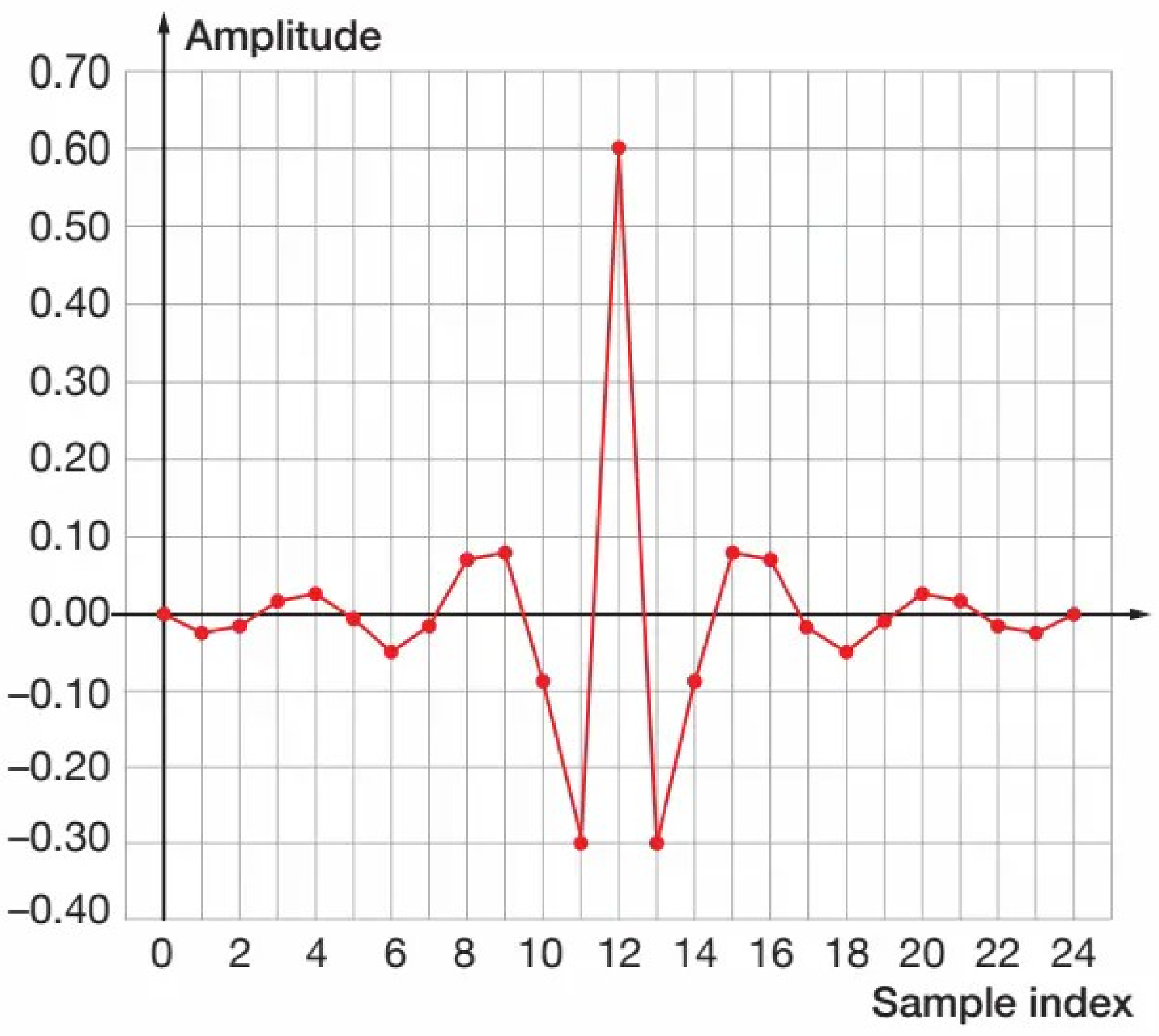


Fig.10: the kernel for a 10kHz high-pass filter obtained by spectral inversion.

then we need to add one to this. Fig.10 shows the coefficients for a 10kHz high-pass filter obtained in this way from the kernel in Fig.8.

High-pass by spectral reversal

Over the range up to the Nyquist frequency (half the sampling frequency, ie, $f_s/2$), spectral reversal is equivalent to shifting the frequency response curve by $f_s/2$. This is shown in Fig.11, which includes the negative frequency part of the frequency response.

Negative frequencies, as related to the mathematical representation of signals and systems, were discussed in *Circuit Surgery*, August 2024. Many circuits and signals have symmetrical responses and spectra in negative frequencies.

In Fig.11, we see that the negative frequency part of the response becomes the high-pass part of the curve after shifting. The gain would drop again somewhere above $f_s/2$, but this is not a concern because our sampled signal will not contain any frequencies in this range – assuming that this is true of the sampled signal, or

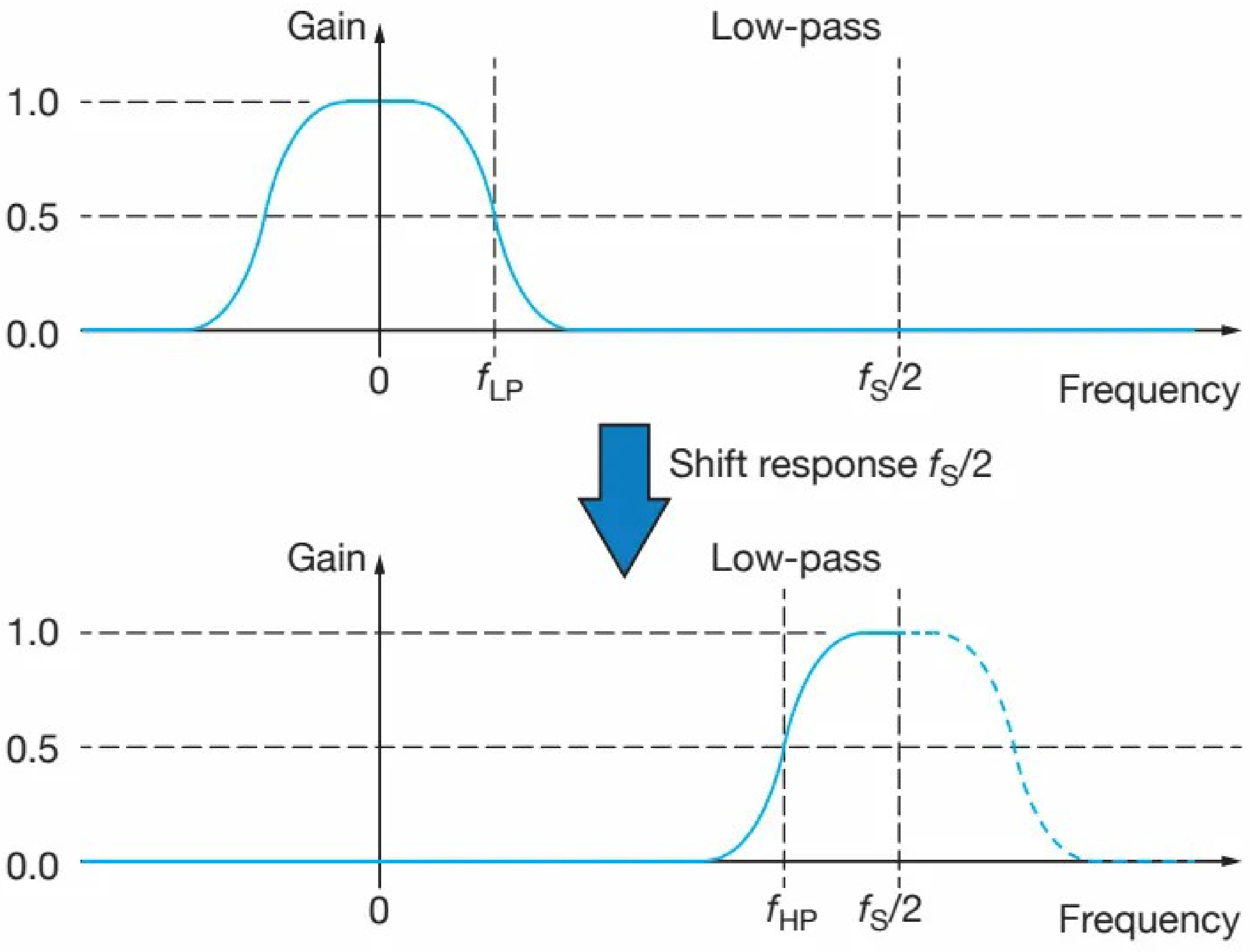


Fig.11: shifting the frequency response to achieve spectral reversal.

that antialiasing filtering has been implemented correctly.

Shifting in the frequency domain by an amount *f* can be achieved by multiplying in the time domain by a sinusoidal waveform of frequency *f*. The most well-known application of this is in modulation and demodulation. For example, an AM radio signal can be produced by multiplying the audio signal by the carrier wave – the audio is shifted to frequencies around the carrier wave frequency.

We discussed this in detail in the *Circuit Surgery* articles on superheterodyne radio starting in December 2023. In radio parlance, the high-pass part of the shifted frequency response in Fig.11 is like the lower sideband.

In the case of the high-pass filter, we are not frequency-shifting the signal, but the response of the filter. This is defined in the time domain by the kernel, so to shift the frequency response of the filter by $f_s/2$, we need to multiply the kernel by a sinusoid of frequency $f_s/2$.

We are working with sampled signals, so we will be multiplying the kernel by

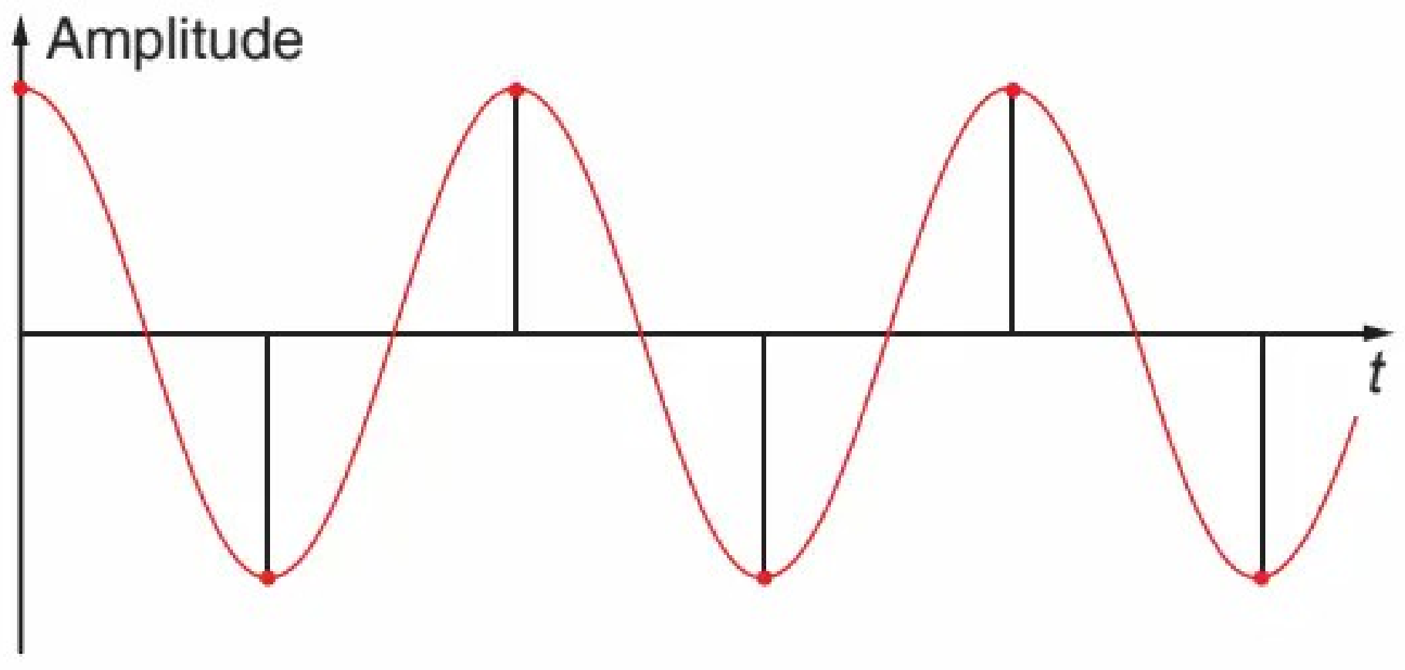


Fig.12: a sinusoid sampled at $f_s/2$.

samples of the $f_s/2$ sinusoid, sampled at the system sampling rate of f_s. For this to work, the samples have to be sensibly aligned – if they were all at the zero crossings, that would not work.

Fig.12 shows a sinusoid of amplitude 1 and frequency $f_s/2$ sampled at the peaks; the resulting sample values are alternating +1 and -1. Any other alignment apart from the zero-crossing case would give the same sign alternation, which could be scaled back to ±1.

So, to obtain a high-pass response, we just change the sign of alternate coefficients of the low-pass kernel, leaving the others as they are (with the middle one unchanged). Fig.13 shows the result of applying this to the low-pass kernel in Fig.8.

A simulated example

Fig.14 shows an LTspice schematic for simulating two high-pass filters (U2 and U3) along with the low-pass filter (U1) from which they were derived. These filters were implemented using the techniques described in recent *Circuit Surgery* articles.

As before, the symbols link to subcircuit definitions in a library file, in which repetitive parts of the code were generated using a spreadsheet, with the coefficients copied from another spreadsheet that was used to calculate their values. The symbols were created by copying the existing symbols and editing the subcircuit name to which they refer.

No other edits were required as the other parameters and I/O connections are the same as before.

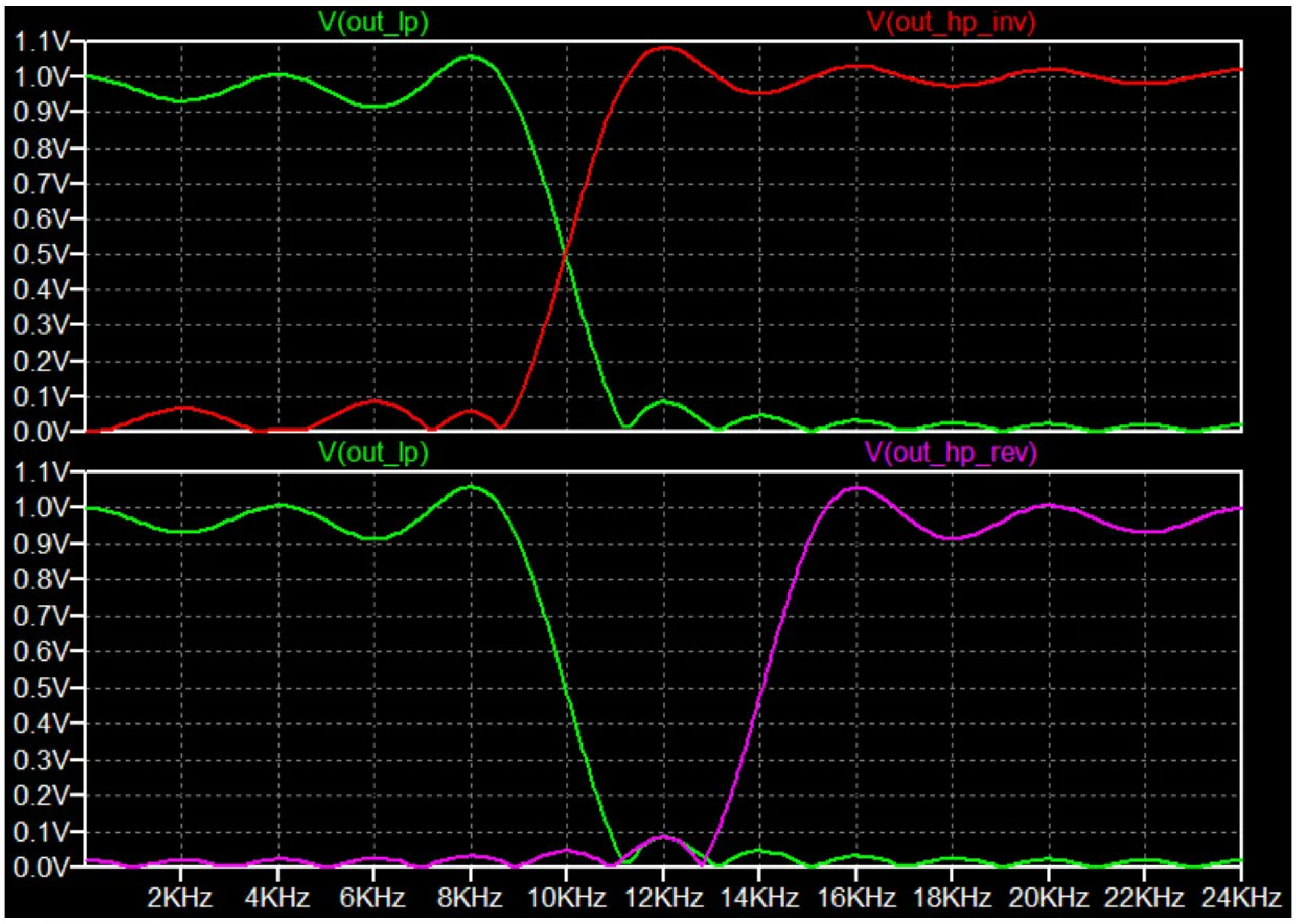


Fig.15: the results from simulating the circuit shown in Fig.14.

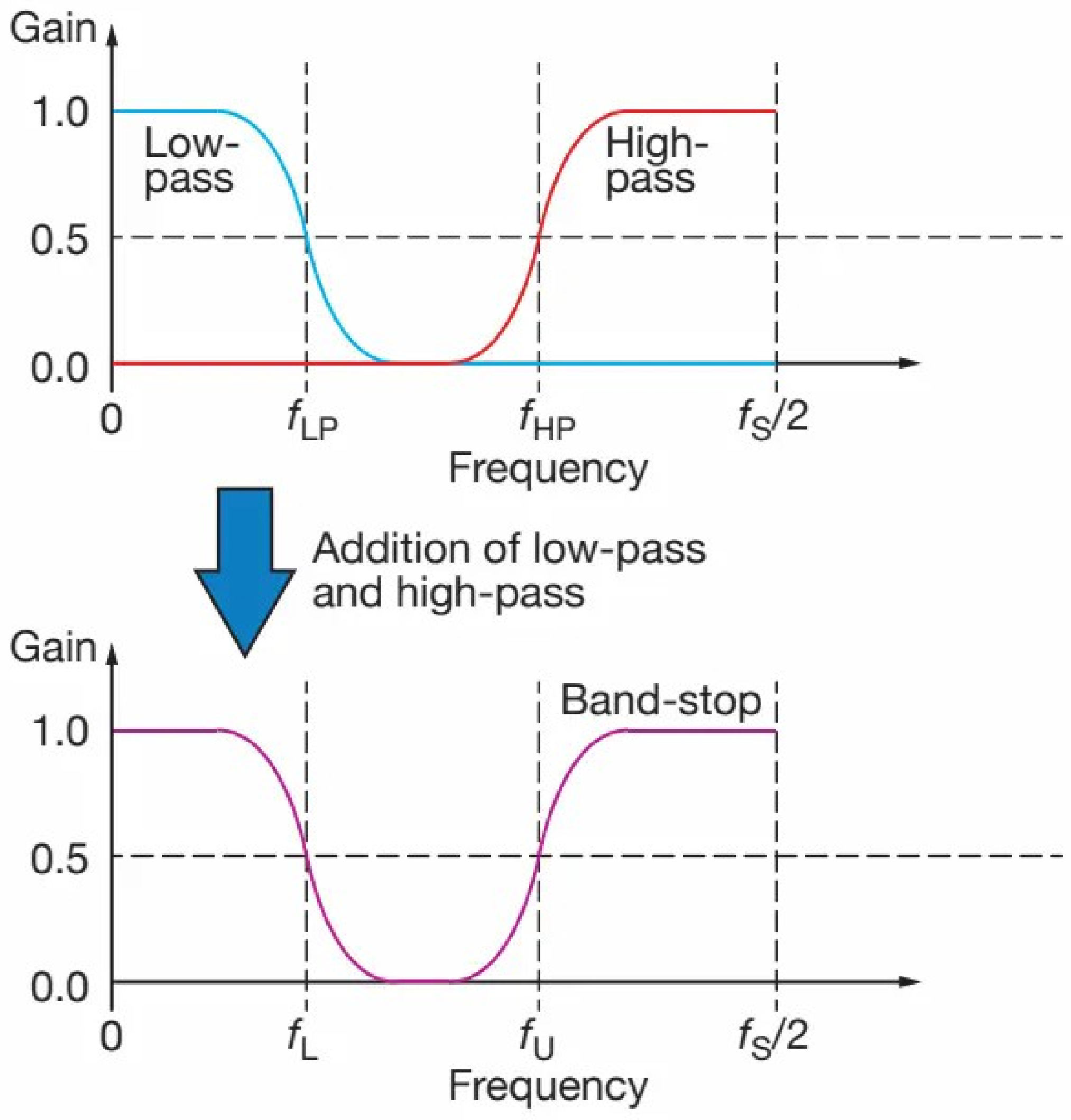


Fig.16: adding low-pass and high-pass responses to obtain a band-stop filter.

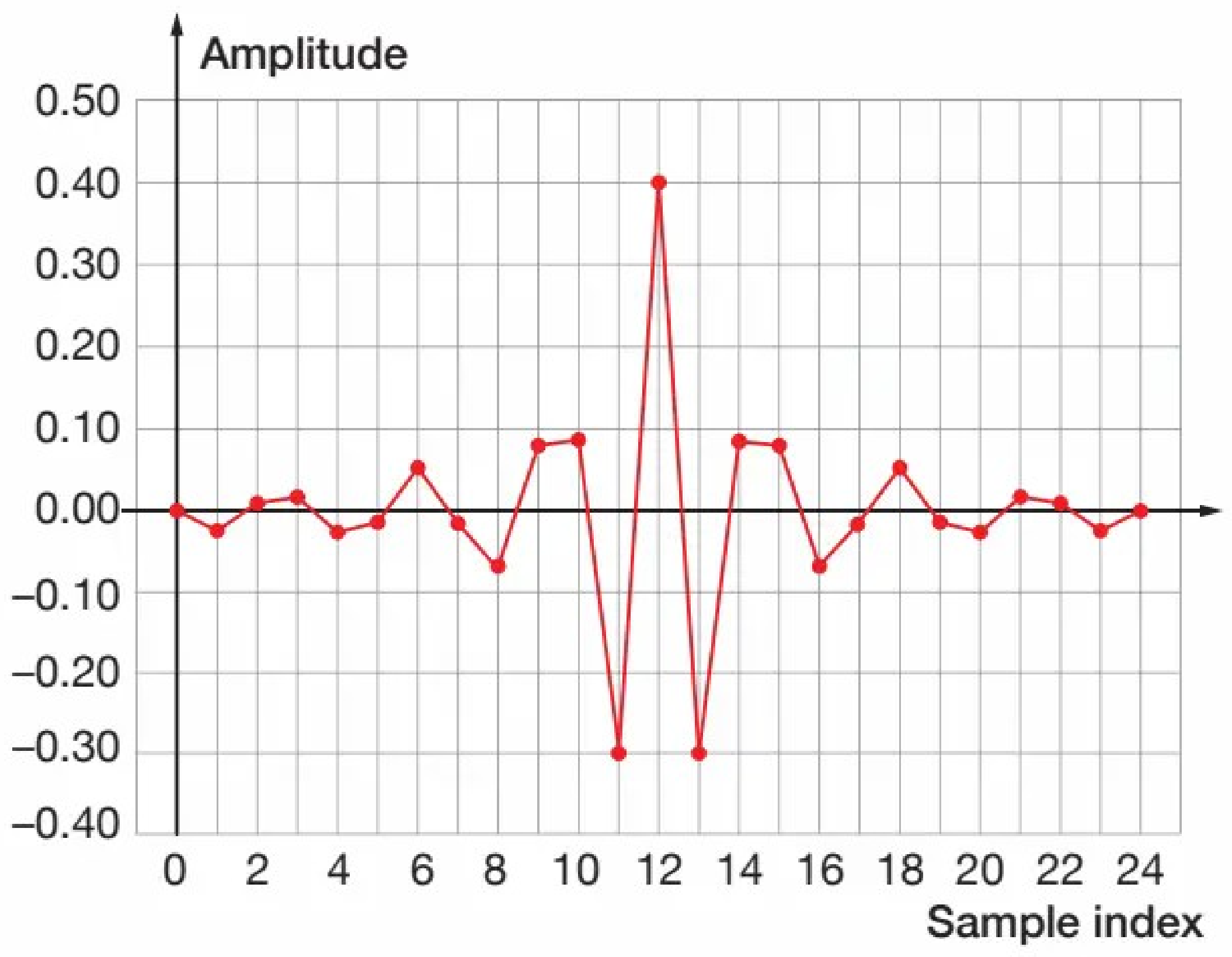


Fig.13: the kernel for a 14kHz high-pass filter obtained by spectral reversal.

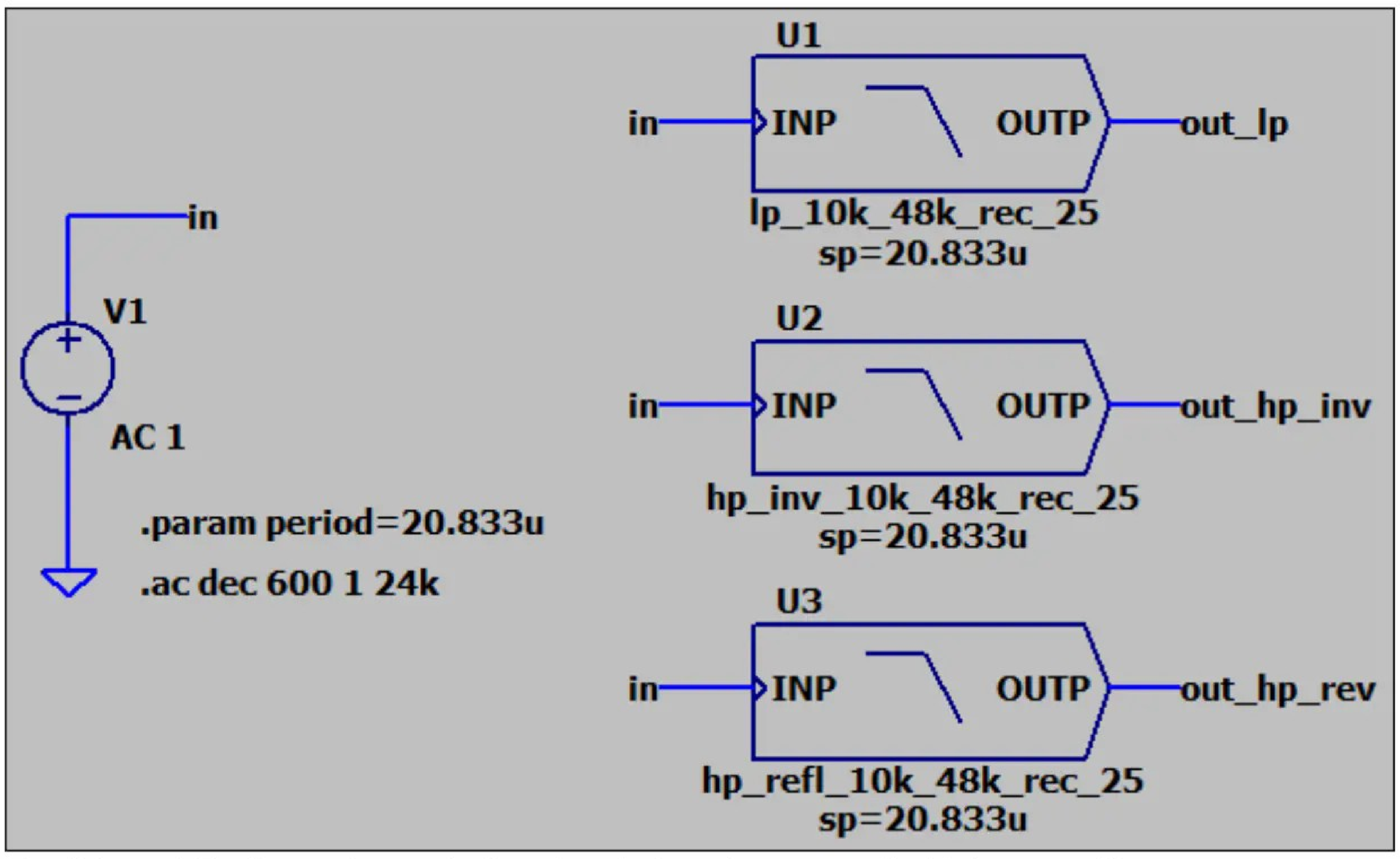


Fig.14: an LTspice schematic for simulating the example high-pass filters.

The filters use 25 coefficients, a 48kHz sampling rate and rectangular windows. The low-pass filter has a cut-off frequency of 10kHz, like the kernel shown in Fig.8.

High-pass filter U2 has coefficients calculated using spectral inversion. In the spreadsheet, this is achieved by adding a column with values equal to the negative of the normalised low-pass coefficients, calculated by the approach described in February 2025. The middle value has a different formula that adds 1 to the negated low-pass coefficient. The kernel is as shown in Fig.10.

High-pass filter U3 has coefficients calculated using spectral reversal. In the spreadsheet, this is achieved by first adding a column in which the rows are alternating 1s and -1s, then adding another column in which the normalised low-pass coefficients are multiplied by these values.

The alternating 1s and -1s can be obtained in various ways, including simple copy-paste, but a better approach is to use the formula $(-1)^i$, that is, minus one to the power of the sample index i. This means the same formula can be used in every row, and the middle coefficient, which always has an even index, will be positive. The kernel is as shown in Fig.13.

Fig.15 shows the simulation results. The upper pane is the frequency response of the low-pass filter (green trace) and the U2 (spectral inversion) high-pass filter (red trace). The U2 high-pass filter has the same cut-off as the low-pass filter, as expected from Fig.3 and the related discussion above.

The lower pane in Fig.15 shows the frequency response of the low-pass filter again (green trace) and the U3 (spectral reversal) high-pass filter (magenta trace). The cutoff frequency of the U3 filter is 14kHz, which is $f_s/2 - f_{lp}$ (24kHz – 10kHz = 14kHz), as discussed above.

Band-pass and band-stop filters

Band-pass and band-stop filters can be obtained by combining low-pass and high-pass filters. We need to consider how the two responses would be combined in the frequency domain and what this implies in the time domain. This requires an understanding of how operations map between the two domains, similar to how spectral reversal used the correspondence between shifting in the frequency domain and multiplying by a sinusoid in the time domain.

Adding suitable low-pass and high-pass functions (where the low-pass cut-off is below the high-pass cut-off) produces a band-stop response, with the resulting stop band being between the two original cut-off frequencies (see Fig.16). This is straightforward to map to the time domain, because addition in one domain corresponds to addition in the other.

Therefore, we can obtain a band-stop filter kernel by adding the coefficients at each index from a low-pass and a high-pass filter (with the same number of coefficients).

Band-pass filters can be obtained by multiplying suitable low-pass and high-pass functions (where the high-pass cut-off is below the low-pass cut-off) – see Fig.17. At frequencies where either of the original filters has a gain of zero, the resulting filter will also have a gain of zero.

Assuming a passband gain of one for both original filters, the gain of the resulting filter will be one where the pass bands overlap. If the transition bands of each filter coincide with the passbands of the other filter, the two transitions of the resulting filter will match those of the original filters (see Fig.16).

As discussed when we introduced convolution in *Circuit Surgery*, August 2024, multiplication in the frequency domain corresponds to convolution in the time domain, and vice versa. Therefore, to obtain a band-pass response from suitable high-pass and low-pass responses, we can perform a convolution calculation on the two kernels.

Unfortunately, convolution is significantly more complicated to implement in Excel than the sign-change and addition operations used in the other response conversions discussed above. On the other hand, if you write code to calculate the kernels (in languages such as C or Python) and have the coefficient data in arrays, the convolution calculation is not particularly difficult.

A convolution can be calculated in Excel by setting out the convolution's reverse, shift, multiply, add operations in a grid, but this is cumbersome for anything but a small number of values.

Another possible approach is to use Excel's LAMBDA functions. These can be used to create your own functions, which become available in a workbook like native Excel functions without needing to use macros with their associated security risks. Some examples can be found online.

Fortunately, it is possible to avoid convolution when finding a band-pass response by performing spectral inversion on a band-stop response. Imagine flipping the band-stop response in Fig.16 upside-down; you get the band-pass response in Fig.17.

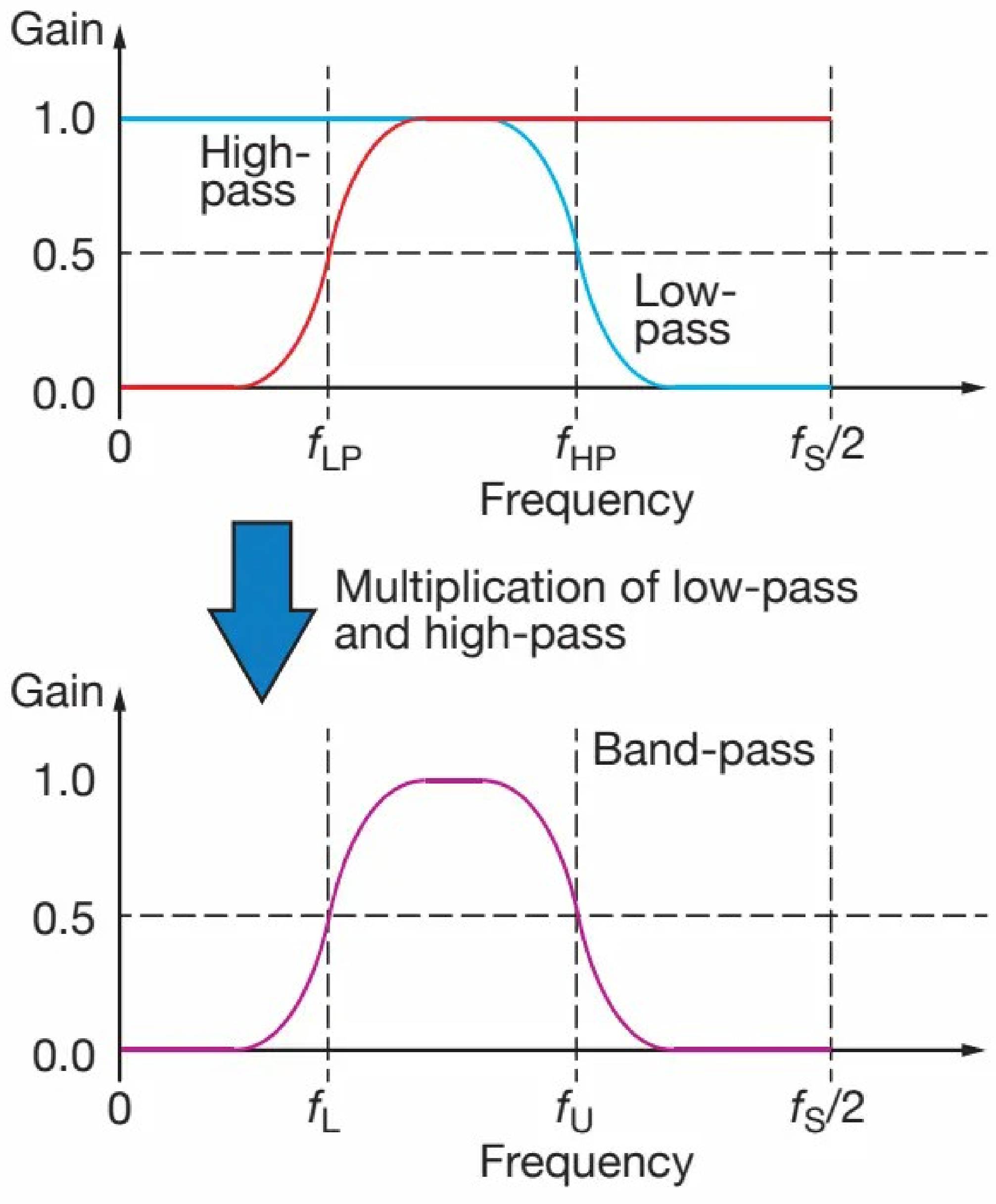


Fig.17: multiplying low-pass and high-pass responses to obtain a band-pass filter.

Therefore, we just need to add two suitable kernels, with the low-pass cutoff below the high-pass cut-off, negate all resulting coefficients and add one to the centre coefficient.

Simulation examples

Fig.18 shows simulation results for 10kHz to 14kHz band-stop and band-pass filters based on the 10kHz low-pass and 14kHz high-pass filters discussed above.

The LTspice schematic (not shown in this article) is very similar to Fig.14, but comprises four filters: the original 10kHz low-pass (output: `out_lp`), the 14kHz high-pass obtained by a spectral reversal of the low-pass filter (output: `out_hp_rev`), the band-stop filter obtained by addition of the low-pass and high-pass kernels (output: `out_bs`), and the band-pass filter obtained by a spectral reversal of the band-stop filter (output: `out_bp`).

After obtaining the coefficients, the new filters were created using the same process as for those in Fig.14.

The upper pane in Fig.18 shows the original low-pass and high-pass filters for reference (green and magenta traces, respectively).

The lower two panes show the band-stop (yellow trace) and band-pass (orange trace) filters, respectively.

The filters used in the examples discussed so far are not particularly high in performance. The filters' transitions are not especially fast, and there are significant ripples in the both the passbands and stopbands.

As examples, this is perhaps helpful in that it makes it easier to see the relationship between the response of the original filters and those derived by manipulating their coefficients.

Still, having said that, it is worth taking a quick look at a higher-performance filter which, as previously discussed, can be achieved using window functions and by incorporating a larger number of coefficients.

Fig.19 (linear) and Fig.20 (dB vs log frequency) show the response of a 10kHz to 14kHz band-pass filter using 129 coefficients and a Blackman window. This was derived from the kernel of a 10kHz low-pass filter, similar to the 5kHz version used in the March 2025 issue.

This filter has an obviously cleaner and sharper response, but it is nowhere near the limit of what can be achieved with digital filters. Using a large number of coefficients with windowing allows very sharp cut-offs to be achieved, which facilitates the use of very narrow stopbands and passbands.

As previously noted, the main disadvantages of the filters we have been discussing is that they have a relatively large input-output delay, and require a relatively large number of calculations.

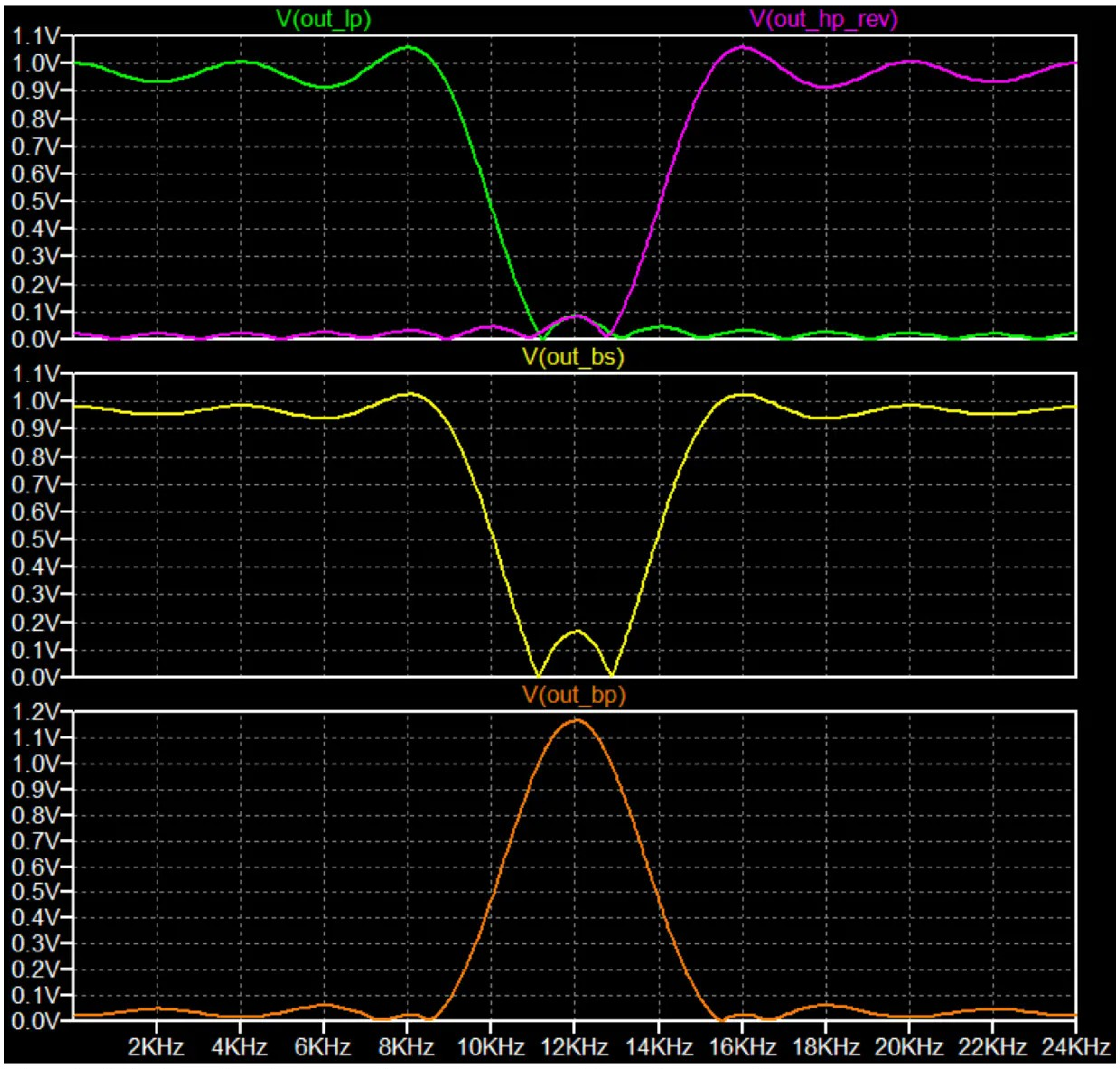


Fig.18: LTspice simulations of band-stop and band-pass filters.

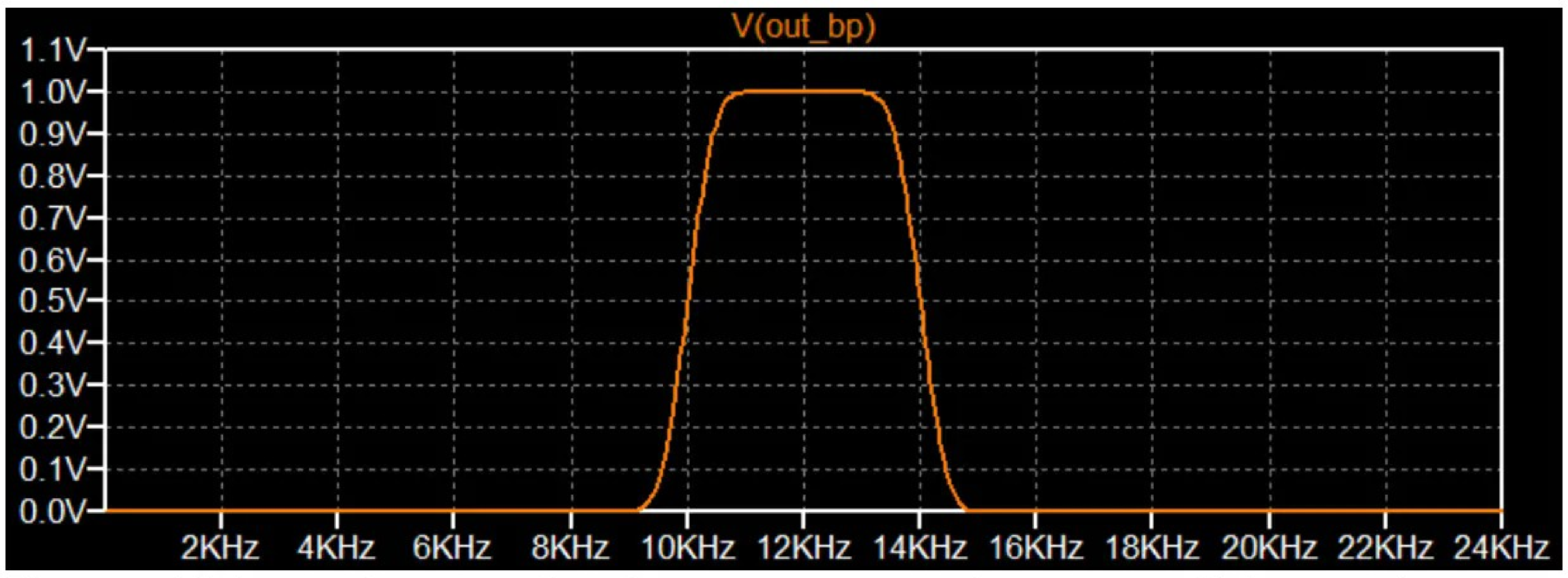


Fig.19: a higher performance band-pass response using 129 coefficients and a Blackman window.

This may mean that a more powerful DSP system is required to implement them than other types of similar digital filters. However, they have the significant advantage that you have very good control over the response, and sharp roll-offs are possible, as are flat passbands.

FIR filters also have some very beneficial properties in terms of phase and delay, which we'll look at in more detail shortly.

Coefficient tools

We have shown how to calculate coefficients for the filters we have discussed.

However, this can be fair amount of work and, as you might expect, there are tools and online utilities that can do the job for you.

If you have access to the MATLAB software, there is a Filter Designer app with wide-ranging capabilities. However, if you are mainly interested in creating the basic types of filter we have been discussing, the FIIIR! online utility from Tom Roelandts (**https://fiiir.com/**) is very useful.

This can calculate coefficients for various types of windowed sinc FIR filters. You specify the sampling rate, cutoff frequency, transition bandwidth and window type, and it computes the filter coefficients for you and plots the responses.

Delay and phase

Earlier, we mentioned the importance of delay in our discussion of calculations for the high-pass filter. The delay

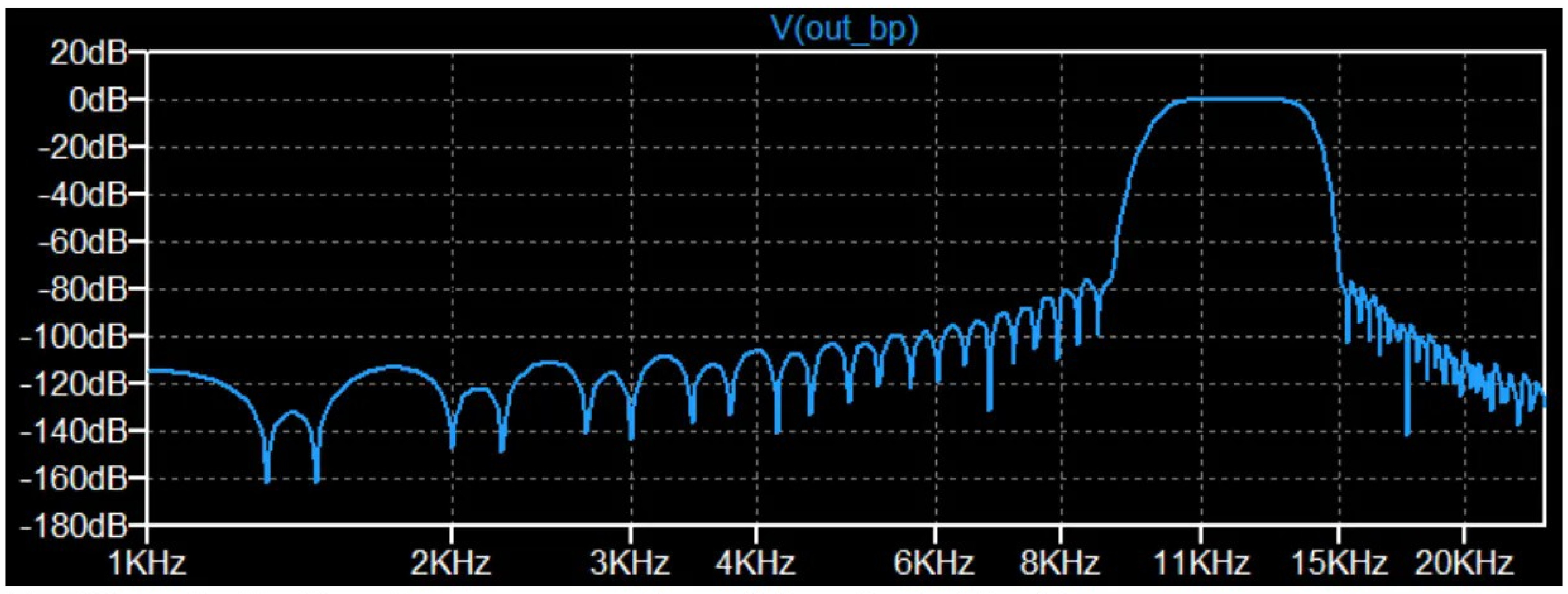


Fig.20: a decibel/log frequency version of the plot in Fig.19.

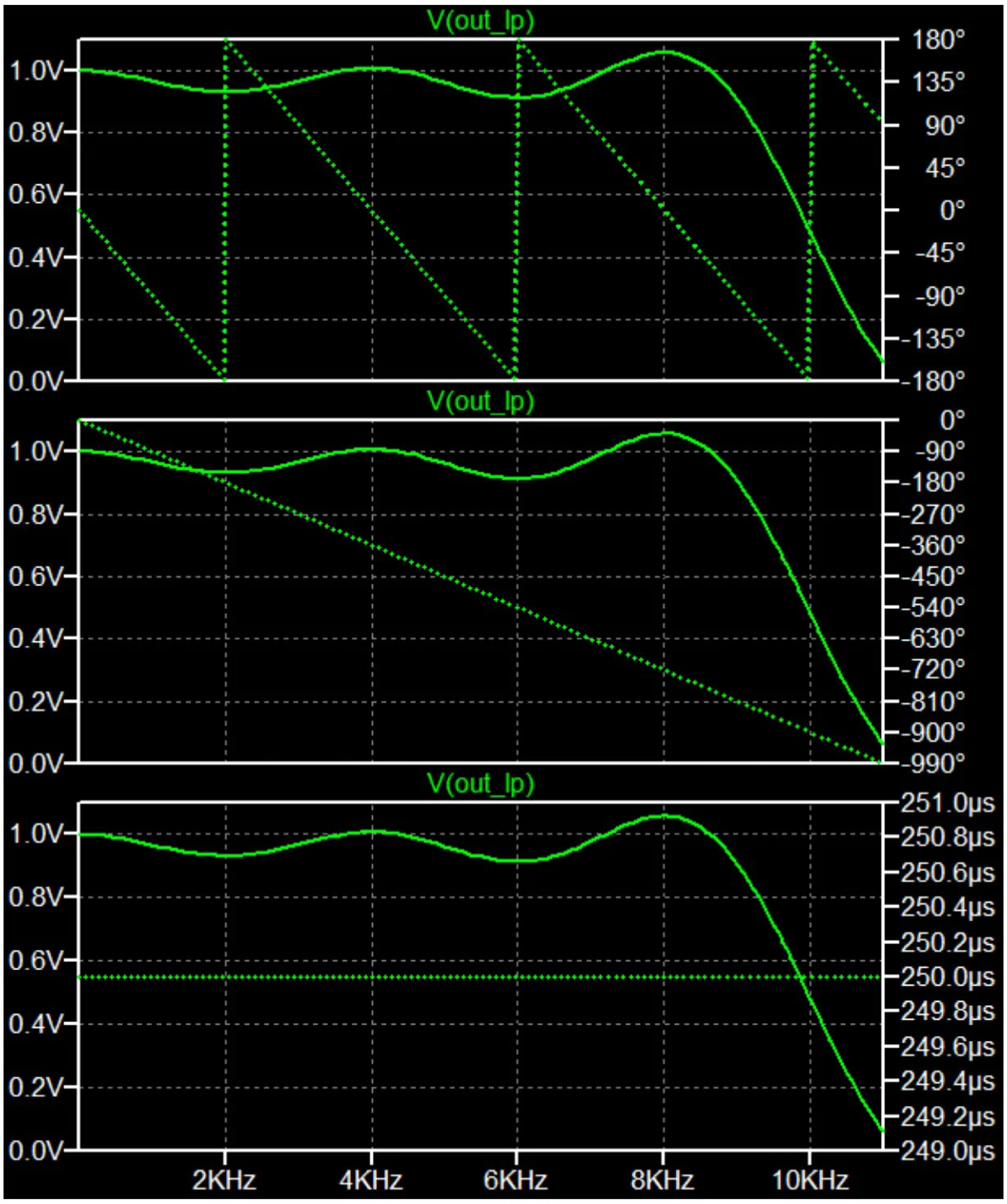


Fig.21: LTspice plots of phase and delay for the low-pass filter from Fig.14.

properties of filters are important, so it is worth looking at this in a little more detail.

When discussing circuits such as filters, it is common to show the delay in the form of a phase shift. The phase shift represents the delay of a circuit in terms of the number of waveform cycles of a waveform (whole or partial, expressed as an angle) at the frequency of interest.

For the filters we have been discussing, we have already noted that the delay is $(N-1)/2$ sample periods for a filter with N coefficients. Our examples have a sample period of 20.833µs (the reciprocal of the sampling frequency of 48kHz).

For the 25-coefficient filters we have used in the majority of examples, the delay from input to output is therefore 12 × 20.833µs = 250µs.

Consider what happens with a circuit that has a fixed delay from input to output as the frequency varies. At 1kHz, our example 250µs delay is a quarter of the signal's cycle time of 1ms. This means that the circuit has a phase shift of -90° at 1kHz (one complete cycle of the waveform is 360°).

Simulation files

Most months, LTSpice is used to support descriptions and analysis in Circuit Surgery.
The examples and files are available for download from the PE website:
https://bit.ly/pe-downloads

The sign is related to the formula for sinusoid, $\sin(\omega t + \Phi)$, where ω is frequency in radians per second and Φ is the phase shift in radians. A negative value of Φ results in the features of the waveform appearing later in time (lagging) compared with $\Phi = 0$ as the waveforms progress.

A fixed delay means that the amount of phase shift will increase linearly with frequency; in this case, -90° for every 1kHz.

At 2kHz, 250µs is half the cycle time of the signal, which is a phase shift of -180°; however, due to the symmetry of the sinewave, this is the same as a phase shift of +180°.

Similarly, at 4kHz, the delay is a complete cycle, which is 360° but equivalent to 0° (no shift) as the shifted and original sinewaves align.

At higher frequencies, the delays correspond to multiple cycles, but will always also be equivalent to a shift within the range ±180°. For example, at 7kHz, the phase shift is 7 × -90° = -630°, which is equivalent to +90°.

The ambiguity in phase values due to its circular nature leads to LTspice having two options for displaying phase. The graphs of frequency response we have shown so far have not included the phase shift, just the gain (magnitude) of the response.

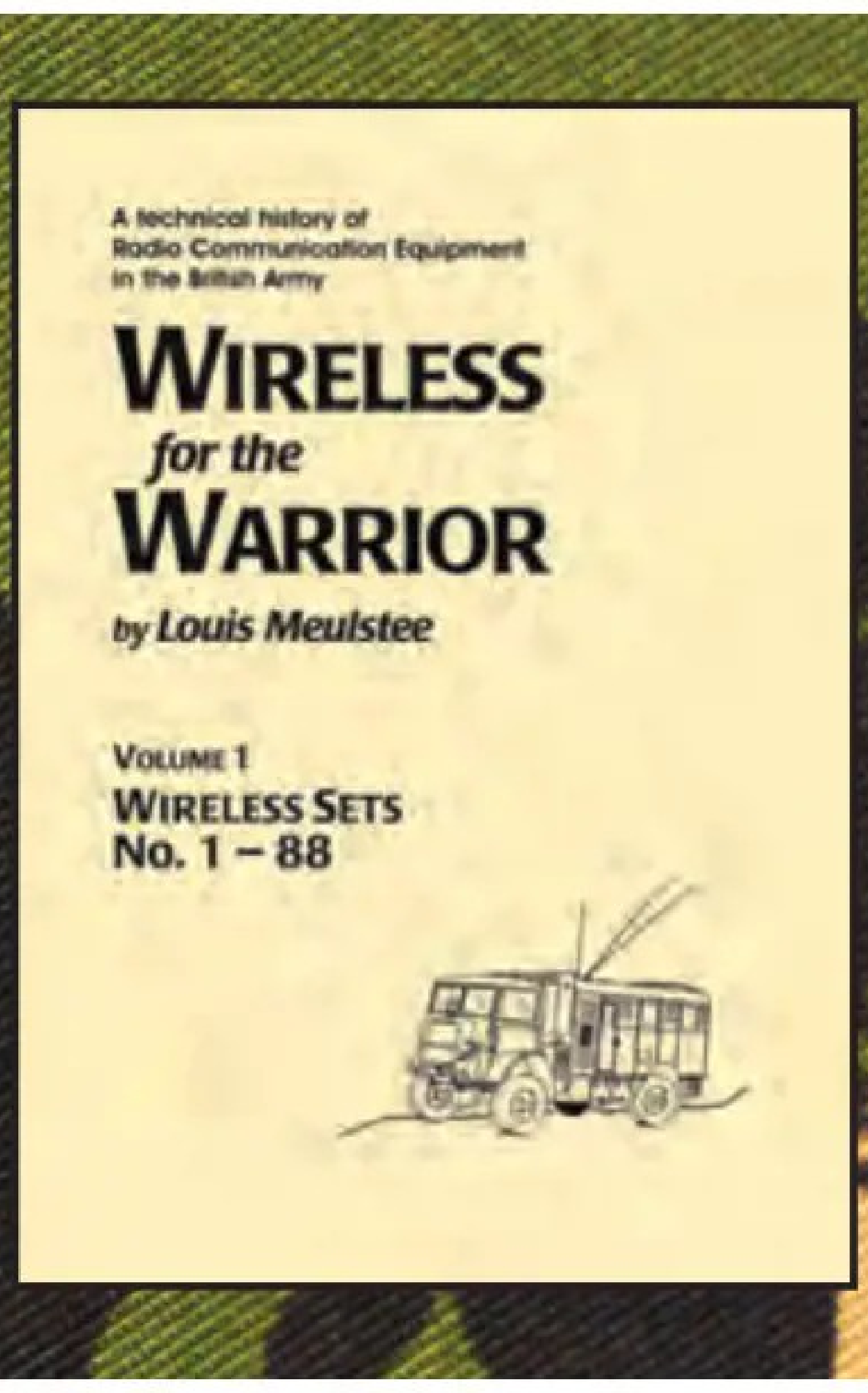
The phase display can be switched on and off by right clicking on the right axis area of the AC simulation result plot, which opens the Right Vertical Axis dialog.

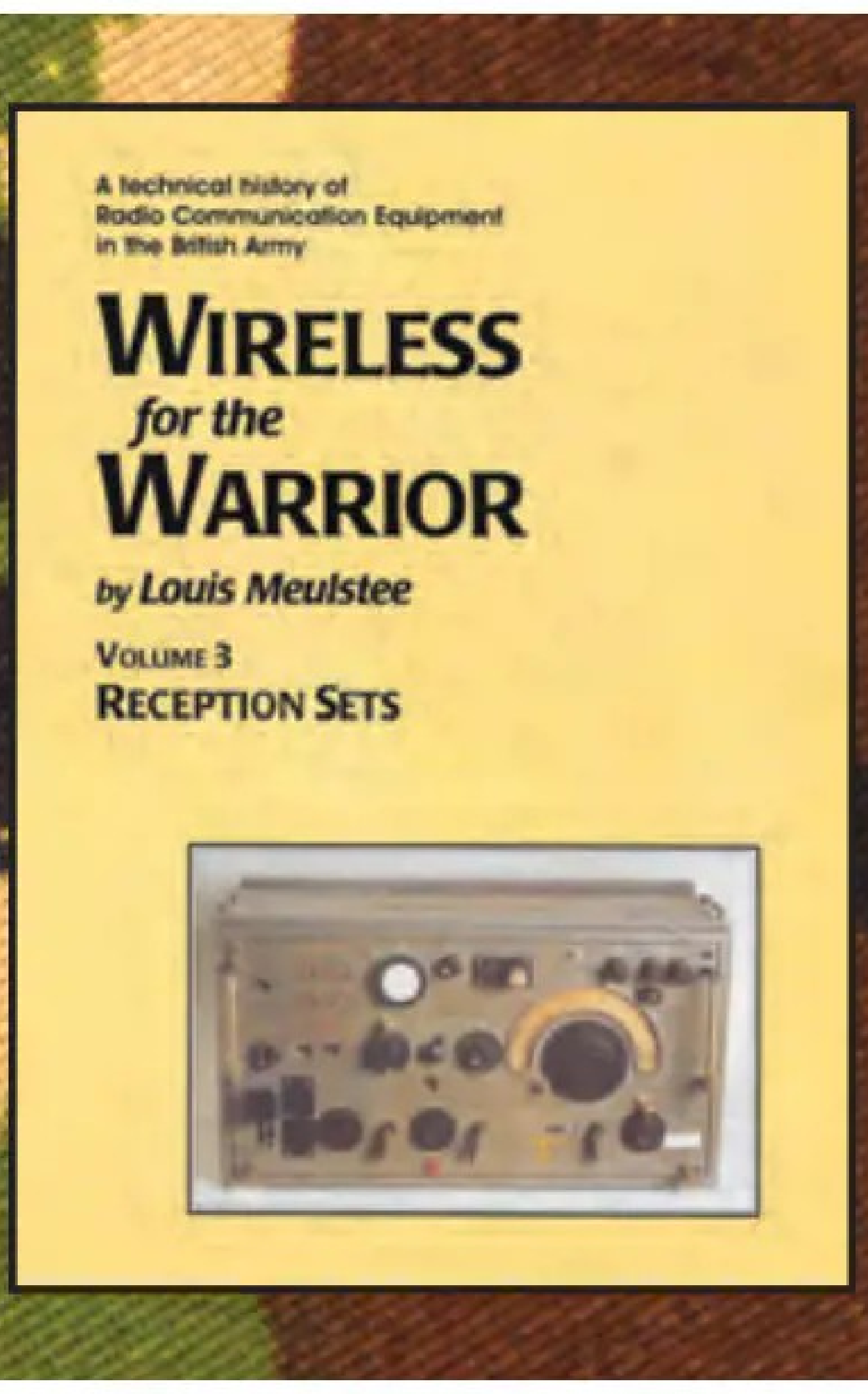
With the phase displayed, if the option "Unravel branch wrap" is selected, the full phase value is plotted; otherwise, it is limited to ±180°.

The delay of the circuit can also be plotted by selecting the "Group delay" instead of "Phase" representation.

The term 'group delay' refers to the delay experienced by a small range (group) of frequencies. For example, this could be the envelope of a modulated signal (eg, the AM signal around a carrier), rather than the simple time delay at a single frequency (referred to as the phase delay).

It is defined as the differential of phase with frequency (ie, the slope of the line on the phase plot). For a circuit with


A technical history of Radio Communication Equipment in the British Army
WIRELESS for the WARRIOR
by Louis Meulstee
VOLUME 1
WIRELESS SETS No. 1 – 88


A technical history of Radio Communication Equipment in the British Army
WIRELESS for the WARRIOR
by Louis Meulstee
VOLUME 3
RECEPTION SETS

The Wireless for the Warrior books are references for the history and development of radio communication equipment used by the British Army from the very early days of wireless up to the 1960s.

Volumes 1 & 3 are still available. Order a printed copy now from:

https://pemag.au/link/ac20

constant delay, the group and phase delays are equal.

Fig.21 shows wrapped phase (top), unwrapped phase (middle) and delay (bottom) for the low-pass filter from Fig.14 for frequencies up to just past the cutoff frequency. The linear increase of phase with frequency and flat delay can be seen and correspond with the values discussed above.

For a signal other than a sinewave, there will be multiple frequency components present. If these are delayed by different amounts, the signal will distort (this is called 'phase distortion'). For this reason, the flat delay/linear phase behaviour of the FIR filters we have been discussing is often a very desirable characteristic. The all-pass filter mentioned above also has a linear phase response.

Fig.22 illustrates the distortion caused by a nonlinear phase/delay that varies with frequency. The top waveform is a pulse containing two frequencies (two added sinewaves with an amplitude envelope). The middle waveform is the result of delaying the two frequencies by the same amount – the shape of the waveform is the same as the input.

In contrast, the lower waveform, where the delays at the two frequencies are different, is clearly distorted (it has a different shape). PE

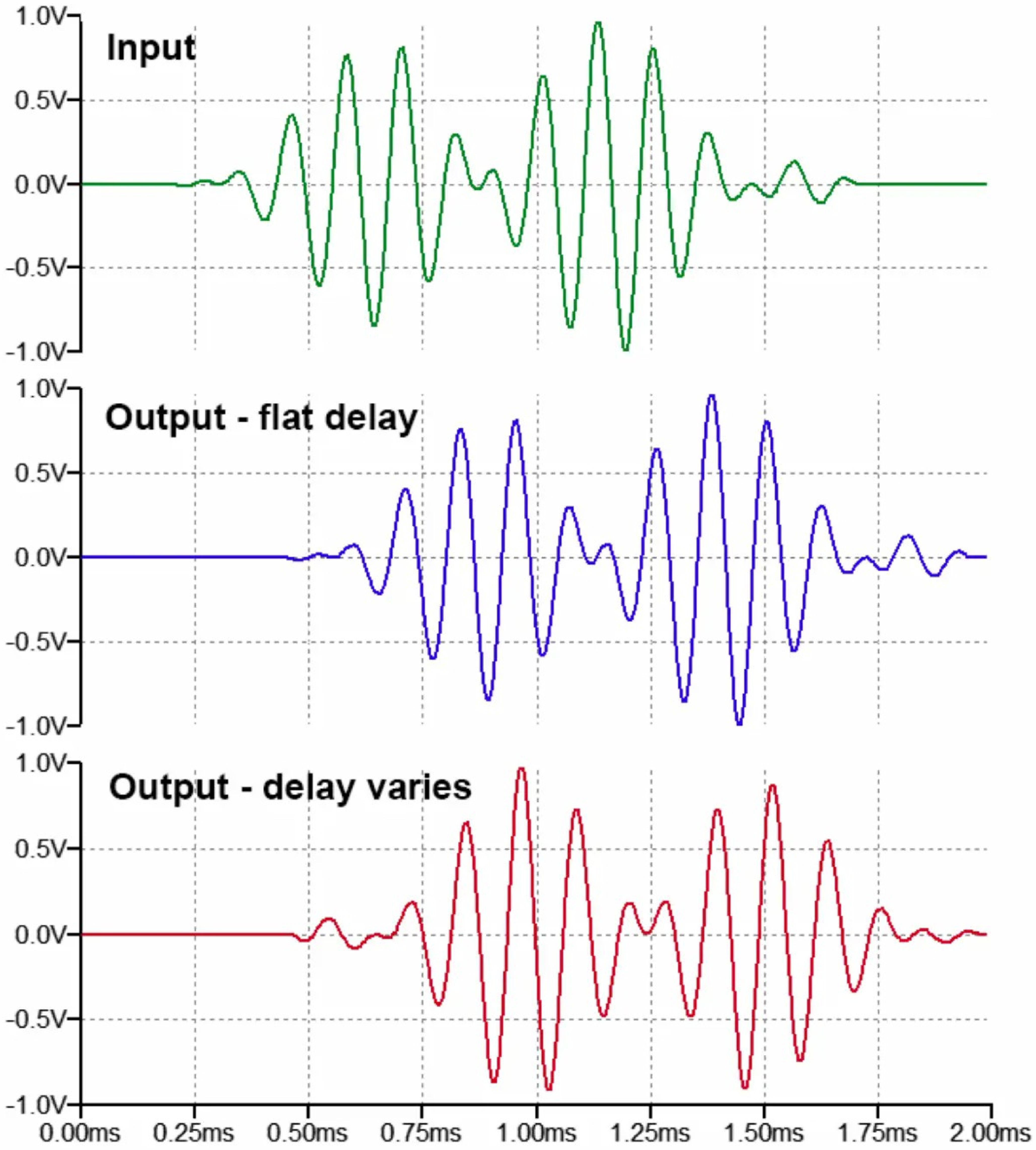


Fig.22: an example of phase distortion.

FIND ALL YOUR ELECTRONIC COMPONENTS IN ONE PLACE

BASIC MICRO UK
ELECTRONICS COMPONENT SUPPLIER
WWW.BASICMICRO.CO.UK

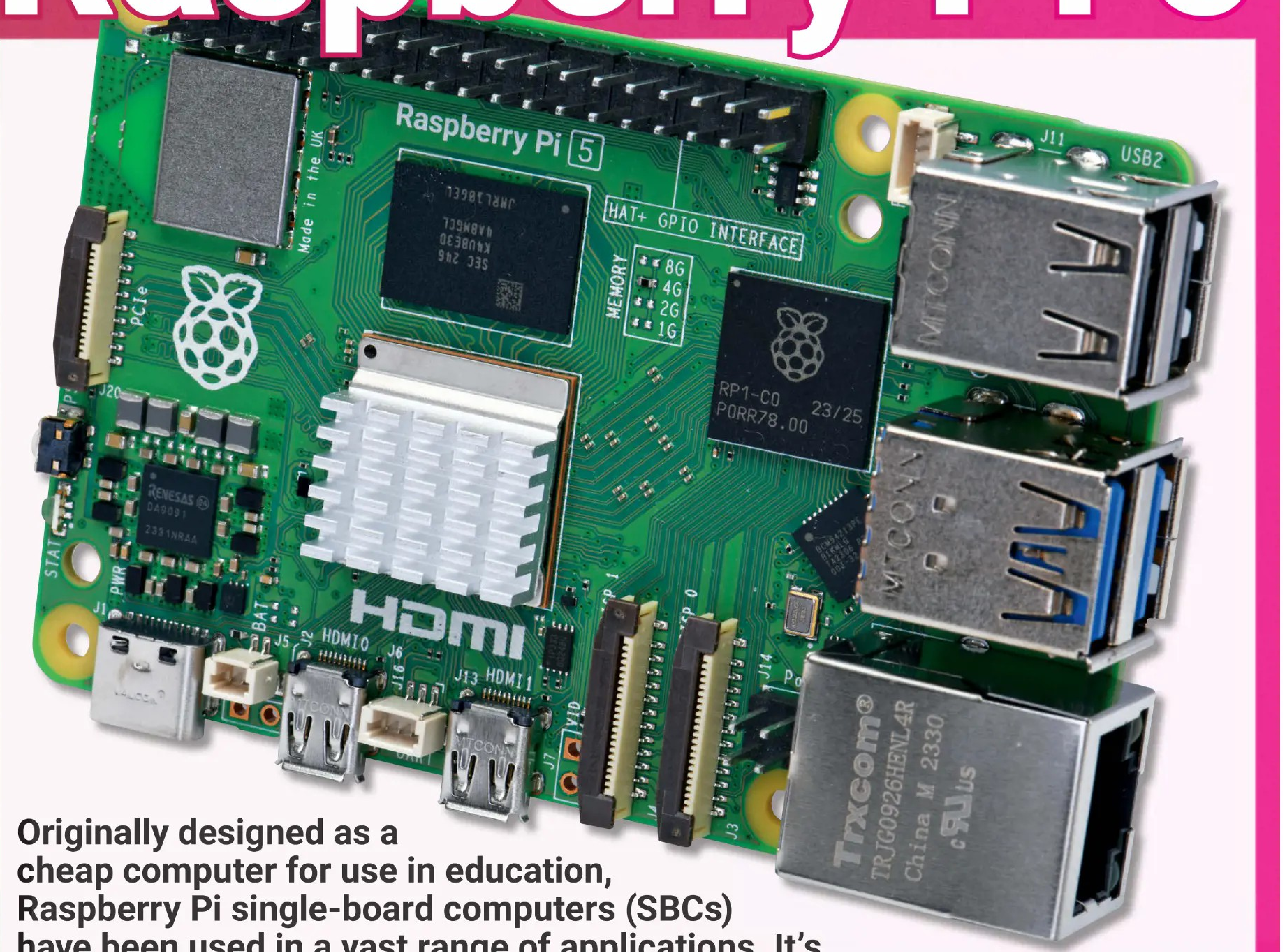
High-quality, genuine parts

01623 720730
Available from 9AM - 5PM Mon-Fri

TRACO POWER
TSR 1-2450

Review by Tim Blythman

Raspberry Pi 5



Originally designed as a cheap computer for use in education, Raspberry Pi single-board computers (SBCs) have been used in a vast range of applications. It's just on five years since the release of the Raspberry Pi 4, and we finally managed to get a Raspberry Pi 5 to test and review.

Since 2012, we have seen the release of a new Raspberry Pi SBC (single-board computer) every year or so. However, there was quite a gap between the Raspberry Pi 4 and the Raspberry Pi 5, which wasn't helped by the component shortages of the last few years.

In 2021, the Raspberry Pi Foundation released the Pico microcontroller board, based on the RP2040 ARM microcontroller, followed by a Pico W variant with WiFi and Bluetooth capabilities. The inexpensive Picos have been embraced by the Arduino, Micropython and Micromite communities.

We have used the various Picos in numerous projects because of their low price and ease of use.

The documentation for the Raspberry Pi Pico is written with the intention of using a Raspberry Pi computer as the development machine. With this in mind and many recent SBCs being touted as replacements for desktop machines, we'll consider the Pi 5's suitability for this task.

2021 also saw the release of the Raspberry Pi Zero 2 W, the most recent iteration of the compact Zero form factor SBCs and the first Zero with a 64-bit processor. It is based on the processor from early versions of the Raspberry Pi 3 but uses a system-in-package (SIP) known as the RP3A0. This combines the processor and RAM into the space-saving package needed to create a Zero board.

The fact that the Raspberry Pi Foundation is now producing its own silicon (both for the Pi Zero 2 W and the Picos) is a notable advance. The Raspberry Pi 5 also includes an RP1 I/O controller, another of their products. We'll delve into the RP1 and other Raspberry Pi 5 features shortly.

The Pi 5

The Raspberry Pi 5 was released in September 2023, with the 4GB RAM variant being available first. There are also now versions with 8GB or 16GB of RAM. Interestingly, the Pi 5 drops

Table 1 – comparison between the ROCK 4C+, Raspberry Pi 4B & Pi 5

ROCK 4C Plus	Raspberry Pi 4B	Raspberry Pi 5	
RockChip RK3399T (6 cores) Dual 1.5GHz ARM-Cortex A72 + Quad 1.0GHz ARM-Cortex A53 1MB + 512KB L2 caches	BCM2711 (4 cores) Quad 1.8GHz ARM-Cortex A72 1MB L2 cache	BCM2712 (4 cores) Quad 2.4GHz ARM-Cortex A76 512kB L2 cache per core 2M L3 shared cache	**Processor (CPU)**
600MHz Mali T860MP4, four shaders, 256KB L2 cache	500MHz VideoCore 6, 1MB L2 cache shared with CPU cores	800MHz VideoCore 7, 2MB cache	**GPU**
two micro-HDMI, up to 4K + 2K (60Hz with one or both)	two micro-HDMI, up to 4K + 4K (60Hz with one or 30Hz for both)	2 micro-HDMI, up to 4K+4K (60Hz with one or both)	**Display output**
HD stereo, up to 24bit/96kHz	Stereo, PWM-based	None	**Audio output**
4GB	1GB, 2GB, 4GB or 8GB	4GB or 8GB	**RAM**
5V/3A, USB-C or pin header	5V/3A, USB-C or pin header	5V/5A, USB-C or pin header	**Power req.**
2× USB2, 2× USB3	2× USB2, 2× USB3	2× USB2, 2× USB3	**USB**
1× Gigabit	1× Gigabit	1× Gigabit	**Ethernet**
802.11 b/g/n/ac (WiFi 5) Bluetooth 5.0 u.FL antenna	802.11 b/g/n/ac (WiFi 5) Bluetooth 5.0 PCB antenna	802.11 b/g/n/ac (WiFi 5) Bluetooth 5.0 PCB antenna	**Wireless**
40-pin header: 1× PWM 2× SPI channels 2× I^{2}C channels 1× ADC (analog) channel	40-pin header: 4× PWM 2× SPI channels 2× I^{2}C channels	40-pin header: 4× PWM 2× SPI 2× I^{2}C	**I/O**

the Model B suffix used for previous models. Given that there was no Model A for the Pi 4, it makes sense that the designations have been streamlined.

We are reviewing the 4GB Pi 5 board. Table 1 shows a comparison between the Pi 4B, Pi 5 and the ROCK 4C+ SBC. The latter is roughly on par with the Pi 4B, although it includes a few nice features that the Pi 4B lacks.

On the other hand, the Raspberry Pi machines have better software support and a larger community. Unsurprisingly, the newer Pi features a faster processor than the 4B. Most benchmarks indicate that the Pi 5 runs at least twice as fast as the Pi 4B. It is an ARM Cortex A76 in the form of a Broadcom BCM2712, which implements the ARMv8.2-A 64-bit instruction set.

Not only is the processor faster, but the microSD card interface on the Pi 5 is capable of running twice as fast as that on the Pi 4B, and the Ethernet interface also transfers data faster. The GPU in the Pi 5 can also drive two 4K displays at 60Hz, compared to the Pi 4B, which can only drive one 4K display at 60Hz.

The main compromises are the power and cooling requirements, with the Pi 5 now specifying a 5V 5A (25W) supply over the Pi 4B's 5V 3A (15W) supply. Our Pi 5 happily booted up with the 3A supply we had been using for our Pi 4B and ROCK 4C+, although it showed a message that 'power to the peripherals will be restricted.' Screen 1 shows the initial desktop with this message.

An official 27W Raspberry Pi power supply offers USB-C PD (power delivery), including 9V, 12V and 15V output voltages. Curiously, the output specified for use with the Pi 5 is 5.1V.

Hardware

Photos 1 & 2 are close-ups of the front and back of the Pi 5 with various features marked out. The overall layout is much the same as earlier models, although it is different enough that it will not fit in cases designed for earlier models. There is little of interest on the back except the microSD card socket.

The main layout difference from the Pi 4B is the transposition of the USB and Ethernet connectors. The mounting holes and GPIO headers are in the same locations, and the other main external features are in much the same, if not identical, positions.

Like the Pi 4B, the USB connector for power input is a USB-C type, and adjacent are two micro-HDMI (HDMI type D) sockets to allow dual monitor connections. The Pi 3B and earlier models have a single full-size HDMI socket and one micro-USB socket.

You'll need a cable with a micro-HDMI plug rather than an adaptor to use both HDMI sockets since the adaptor will likely foul the USB-C socket. Our basic single-monitor setup worked using the HDMI socket (with an adaptor), HDMI1, further from the USB-C socket.

The top of the Pi 5 looks quite sparse; many of the passive components are on the back of the board. The main processor is the larger chip with a metal shield (we attached an aluminium finned heatsink to it, visible in the photos); the rectangular chip next to it is the RAM. The second shielded package is the radio module, providing WiFi and Bluetooth connectivity.

The RP1 'southbridge' I/O controller is the large chip with the Raspberry Pi logo near the USB sockets. This is one of the ICs the Raspberry Pi Foundation has designed and produced. The RP1 connects to the processor via a four-lane PCIe 2.0 interface.

Bundling many of the I/O functions into a single chip allows substantial performance improvements for the Pi 5 over the Pi 4B. The RP1 even handles GPIO functions on the 40-pin header and has been designed

Raspberry Pi 5

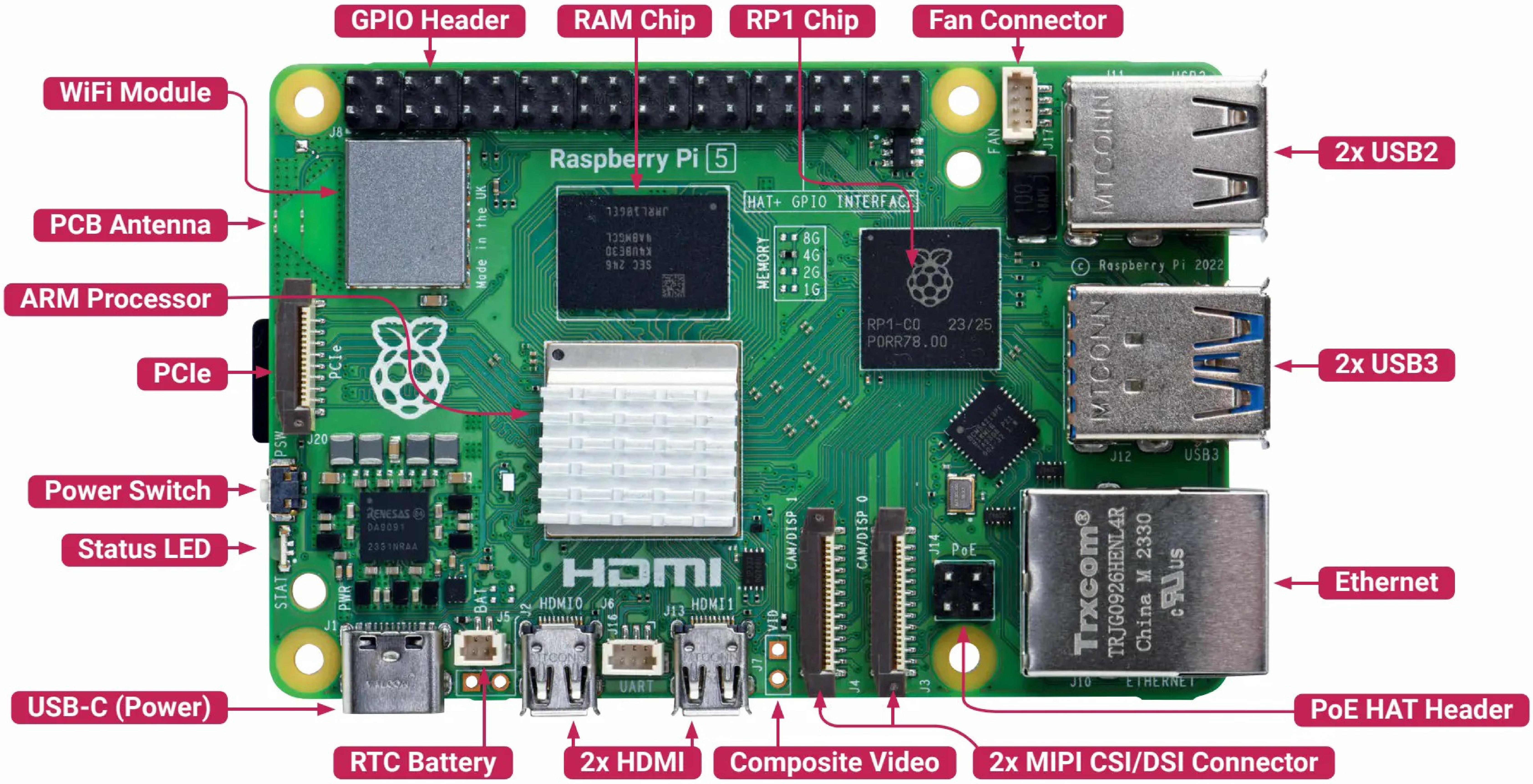


Photo 1: the Raspberry Pi 5 is the same size and shape as its predecessors, but the connectors have been slightly rearranged, so it requires a different case. The official case includes a small fan that provides much-needed cooling. The same GPIO pinout applies as the previous Pis, so most existing HATs should work with the latest Pi. The supplied RAM is indicated with a component fitted to the MEMORY box.

to provide the same I/O functions as the Pi 4B.

The RP1 provides Gigabit Ethernet, two USB 3 interfaces, two USB 2 interfaces and two MIPI transceivers for cameras/displays on the J3 and J4 CSI/DSI connectors. The RP1 also includes the versatile PIO (programmable input-output) peripheral and an ADC (analog-to-digital converter). These latter two features are not used on the Pi 5.

The RP1 relieves the main processor of most of the peripheral duties. There is more information on the RP1 at **https://pemag.au/link/abvc**

The Pi 5 dispenses with the 3.5mm TRRS socket used for audio and composite video in earlier versions. Instead, video is available from a dedicated two-pin header (marked as VID next to HDMI1). Two of the GPIO pins on the 40-pin header can produce PWM-based audio, although this does not appear to be enabled by default.

The top of the board also has a four-pin PoE (Power-over-Ethernet) header for connecting to a PoE HAT. HAT (hardware attached on top) is the Raspberry Pi terminology for a shield or daughterboard.

The top of the Pi 5 also breaks out a four-pin polarised header (J17) for a fan. An active cooler is available to suit the Pi 5, which can connect to J17. The active cooler mounts to two holes adjacent to the four main mounting holes. The official case for the Pi 5 also incorporates a fan that can be powered from J17.

A three-pin polarised header (J16), labelled UART, can be used for diagnostics.

The Renesas DA9091 PMIC (power management integrated circuit) is near the USB-C socket. It incorporates a real-time clock (RTC) feature that utilises an optional battery connected to the nearby J5 polarised header.

The two-pin header pads marked J2 next to J5 are connected in parallel with a momentary pushbutton (marked PSW) used as a power switch. It is adjacent to a bicolour LED labelled STAT.

J20 is a flexible flat cable (FFC) connector marked as PCIe that breaks out a single PCI Express 2.0 lane. Future

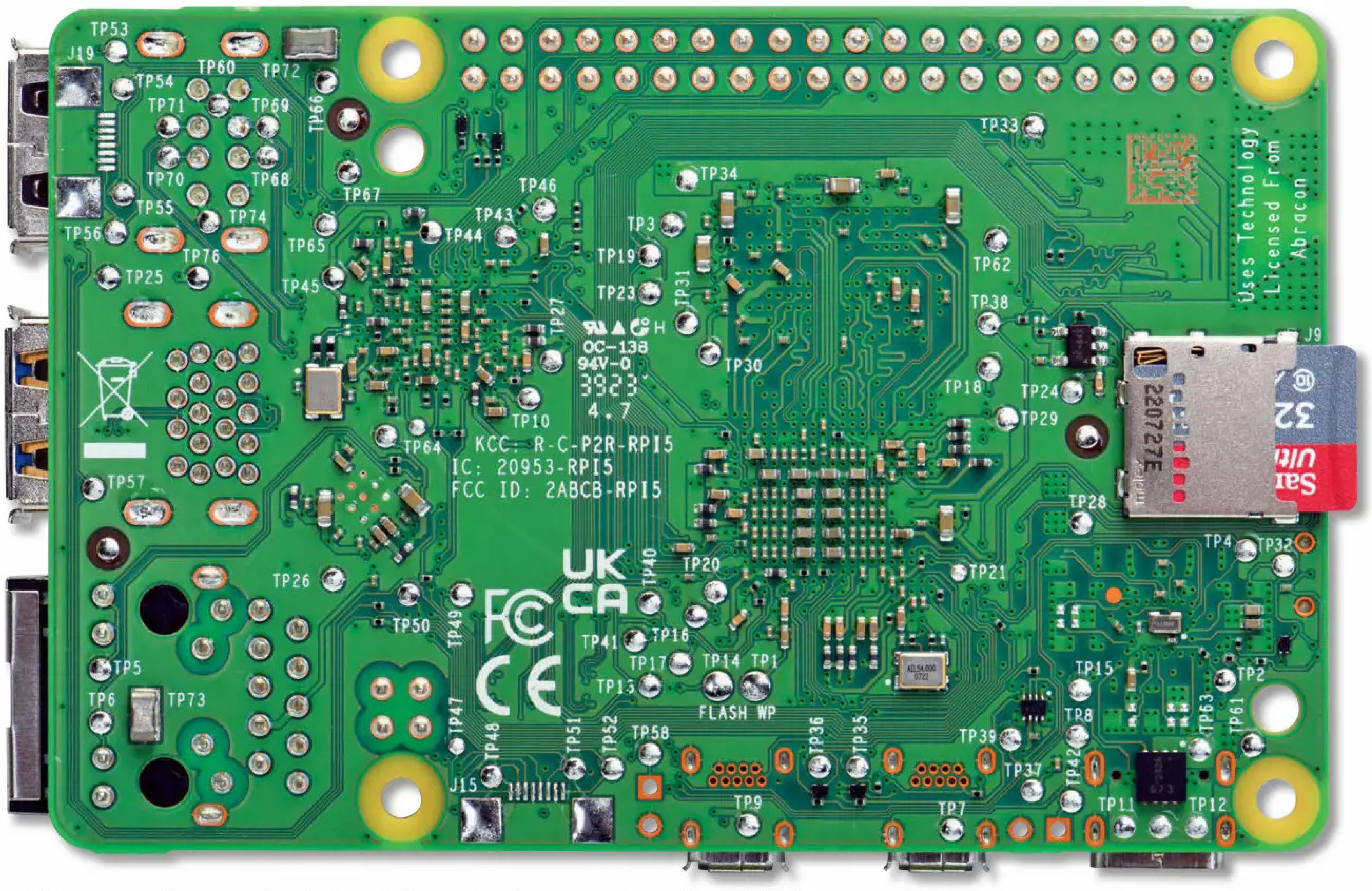


Photo 2: the underside of the Pi 5 is populated mainly by passive components.

HAT designs will use this interface; in fact, the Raspberry Pi M.2 HAT+ has been announced, which will allow the connection of M.2 peripherals such as NVMe drives.

The back of the board is mainly populated with passive components and the microSD card socket that holds the operating system. There are also options to configure a boot EEPROM to allow booting from a USB storage device or an NVMe SSD.

Setting it up

Like just about every other SBC, the Pi 5 typically uses a microSD card for the operating system and user files. Hence, installation involves transferring a disk image to the card using another computer. The Raspberry Pi Foundation provides the Raspberry Pi OS, which is based on Debian Linux.

Operating system downloads can be found at **https://pemag.au/link/abvd** and that page indicates which versions are compatible with which Pi boards. There are bundles pre-loaded with different programs. We used the latest version at the time of writing (v5.2), which includes all the recommended software.

This download comes to around 3GB and expands to a 15GB file. A 32GB card is recommended. We previously used WinDiskImageWriter to transfer the image files to the microSD card, but this time, we tried Raspberry Pi Imager, which has been available since 2020. This, as well as other software, can be downloaded from **www.raspberrypi.com/software**

Screen 2 shows the Imager program. It can automatically download card images as well as write previously downloaded files. Imager can also configure the image with settings like WiFi, country and SSH, allowing the Pi to operate in headless mode (without a keyboard, mouse or monitor). Writing the file and verifying the image took about half an hour; the verification is a nice touch.

The Imager is a good way to see what other software is available. It lists media player and emulation images, among others. Even if you don't have a Pi, we suggest downloading Imager to see what other people are doing with their Pi.

Once the image is transferred, the Pi 5 is booted by installing the card, connecting the monitor, keyboard, and mouse, then plugging in the power supply. The first boot sets up a few things and performs a system update.

Once everything was set up and the update completed, the Pi 5 responded quickly. A reboot took about 15 seconds, comparable to modern computers fitted with SSDs.

Using it

The Raspberry Pi Foundation does a good job of making their software easy to use; the mix is much the same as earlier distributions. Educational programs like Scratch, Mathematica and Wolfram are included, as is Thonny (an integrated development environment [IDE] for the Python programming language). All these programs would be familiar to seasoned Pi users.

We then looked for programs that would be useful in a typical office environment. The LibreOffice suite (including word processor and spreadsheet) was installed, as were the Chromium and Firefox web browsers. Many of the included programs may not be familiar if you have previously only used Windows or macOS. However, they will be known to those familiar with open-source alternatives to proprietary programs.

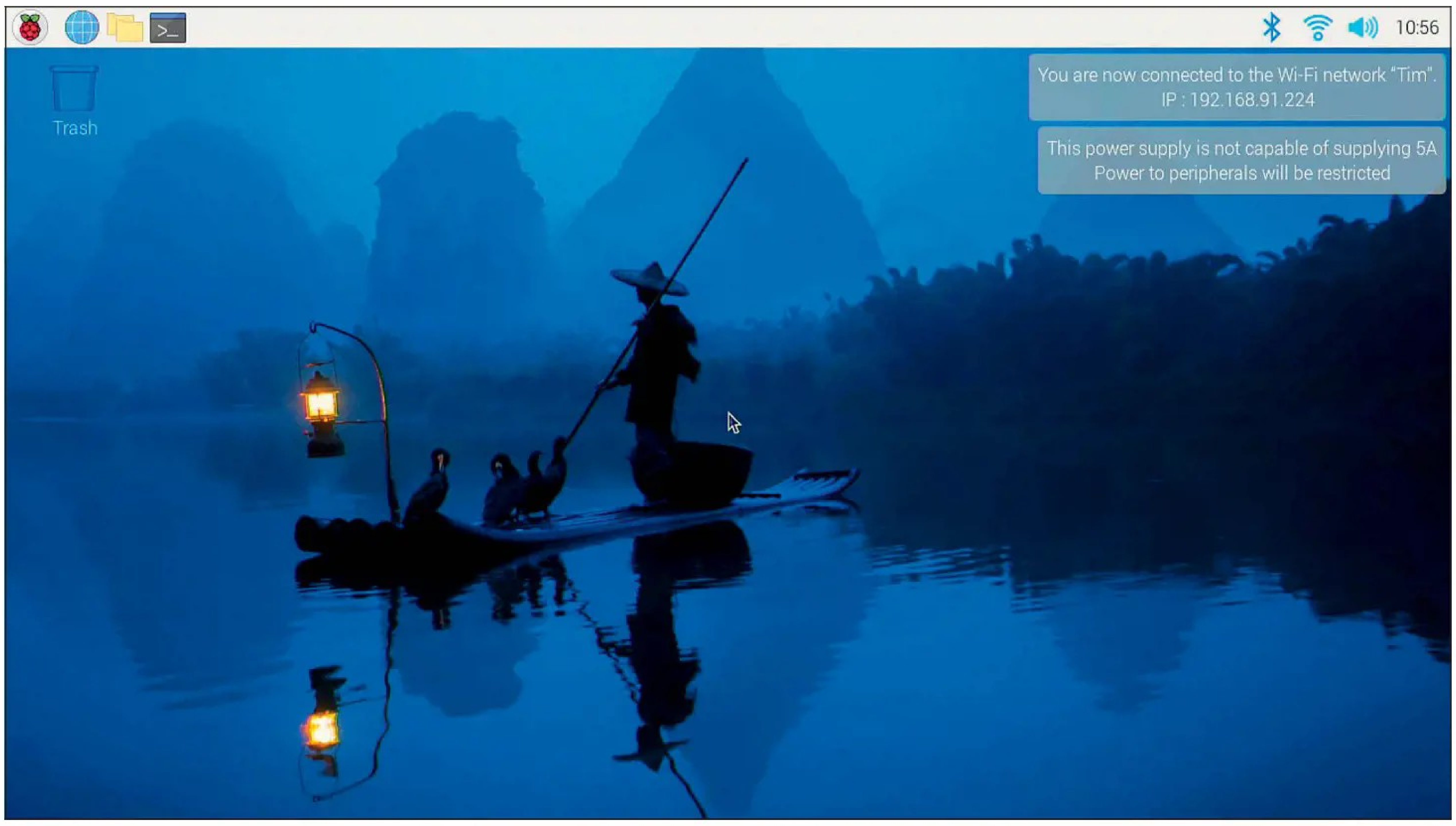
Even the open-source KiCad EDA (electronics design automation) suite is installed. The Arduino IDE is not installed by default, but it and many others can be added through the Preferences → Add/Remove Programs dialog box. Using the Arduino IDE on the Pi 5 was practically the same as on the Windows machines we are used to.

Some programs we use, like Altium Designer, are only available for Windows operating systems. Although the MPLAB X IDE is available for Linux (and Raspberry Pi OS is a Linux variant), currently, it only works on x86 and x64 processors and not ARM processors.

We were able to program a Pico from the Pi 5 from a command line interface with relative ease.

So, a good proportion (but not all) of the programs we use daily are available or easy enough to install on the Pi 5. ARM processors are becoming more common on portable and desktop computers, such as M2-based Mac computers or Microsoft Surface devices with an SQ2 processor.

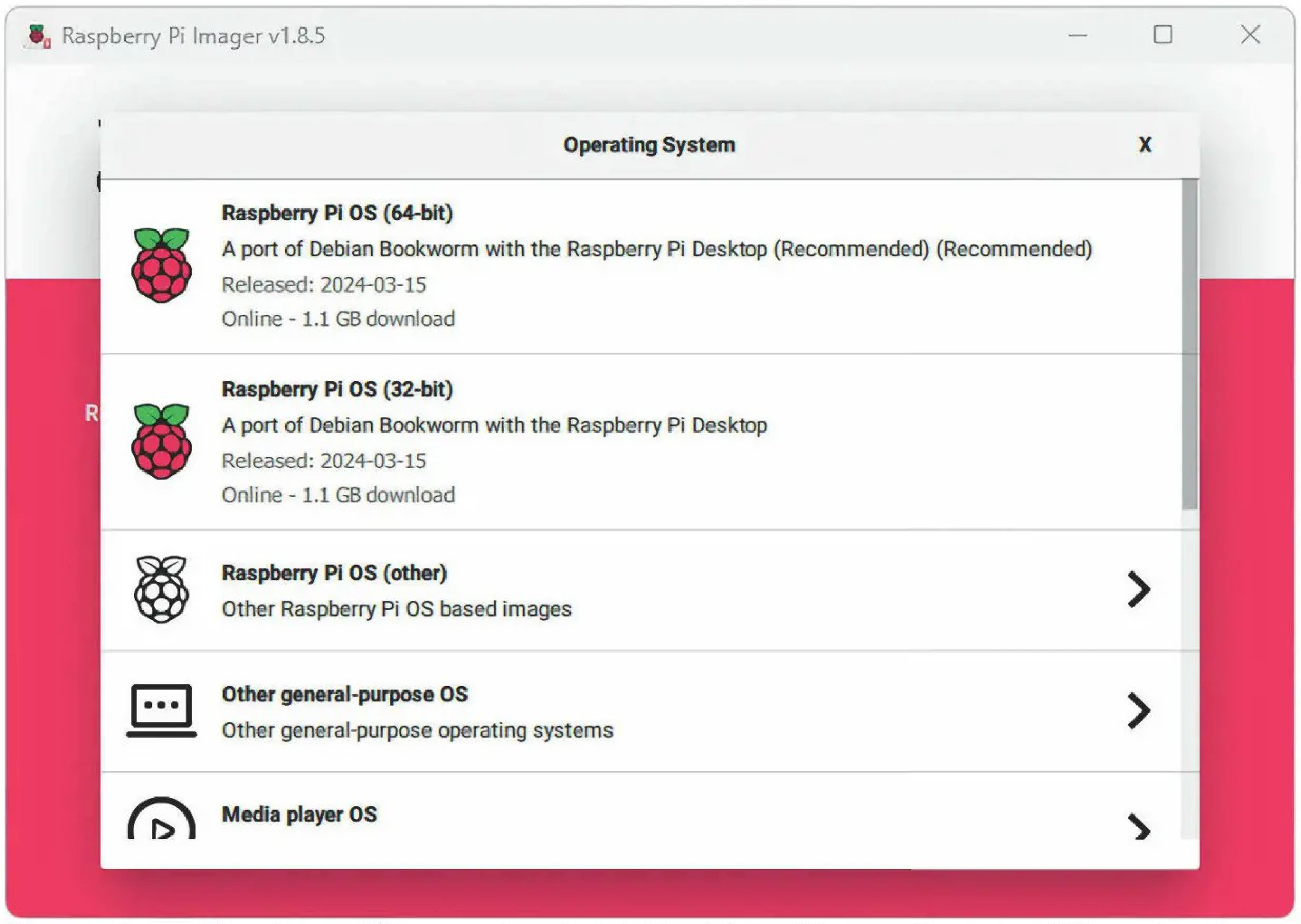
We expect support for ARM processors to grow steadily; hopefully,



You are now connected to the Wi-Fi network "Tim".
IP : 192.168.91.224

This power supply is not capable of supplying 5A
Power to peripherals will be restricted

Screen 1: the initial desktop after setting up the Pi 5; it looks much the same as previous versions. The messages at top right indicate that it has connected to a preconfigured WiFi network and that the connected power supply cannot provide the 5A needed for full functionality.



Raspberry Pi OS (64-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop (Recommended) (Recommended)
Released: 2024-03-15
Online - 1.1 GB download

Raspberry Pi OS (32-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop
Released: 2024-03-15
Online - 1.1 GB download

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

Other general-purpose OS
Other general-purpose operating systems

Media player OS

Screen 2: the Raspberry Pi Imager is a helpful tool for setting up the microSD card and seeing what other disk images are available. Initially, we ran this on a Windows computer but it comes preinstalled on the Pi.

that will translate to better software options for computers like the Raspberry Pi.

The performance of the Pi 5 was generally quite good, and the system seemed responsive. The processor gets very hot, though; too hot to touch, so one of the cooling options would be beneficial.

Raspberry Pi Desktop

An interesting footnote we found on the www.raspberrypi.com/software/operating-systems page is Raspberry Pi Desktop. It's an operating system image for PC and Mac computers (those with x86 or x64 processors) that provides a Linux environment similar to that found on the Raspberry Pi boards.

We loaded this onto a USB flash drive with a program called Rufus (https://rufus.ie/en/), which is a utility that can be used to create bootable flash drives. We plugged the drive into an older PC and booted it up.

The flash drive can install the Raspberry Pi Desktop operating system to the hard drive (so you don't need to boot from the flash drive). Alternatively, you can run it directly from the flash drive. Screen 3 shows the desktop environment and program installation.

The Raspberry Pi Desktop is based on Debian 11, an older version than the Debian 12 used in current versions of Raspberry Pi OS (for the Pi SBCs). Still, the appearance and functionality are similar.

The Desktop software lacks broad hardware support, so we couldn't fully use the PC's features. In particular, WiFi would not work, so we had to devise an alternative way to connect to the internet using a USB dongle.

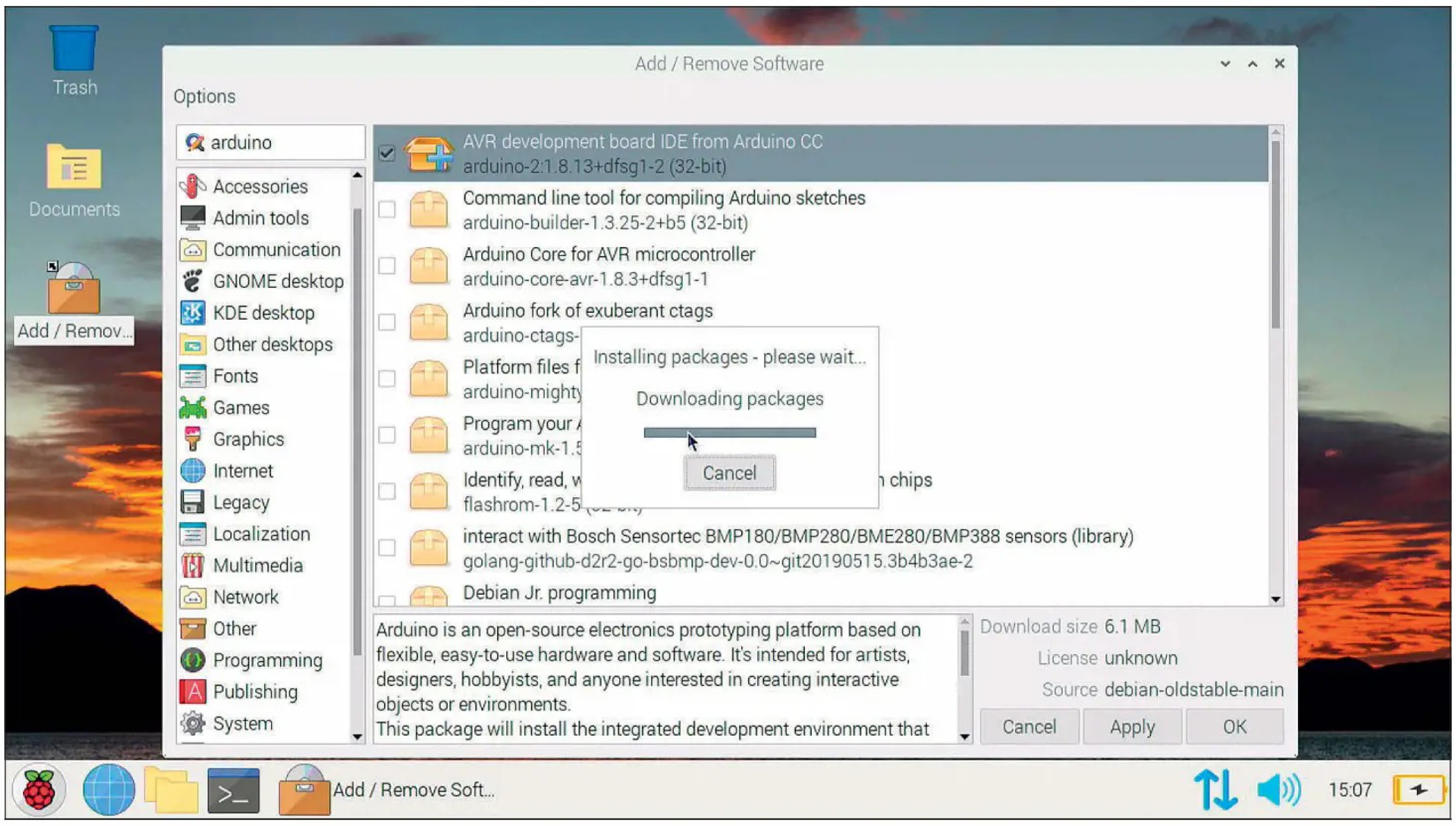
If you have an old PC, Raspberry Pi Desktop could be an easy way to try out the Raspberry Pi OS. Be aware that the flash drive and your PC's hard drive could be erased if you do that.

Conclusion

With ARM chips gaining a foothold in the market traditionally held by x86 and x64 processors, software availability for computers like the Raspberry Pi can only grow. The Raspberry Pi Foundation is now producing some of its own chips; that's a promising sign, and we look forward to their future developments.

While it's still no match for most PCs, the Raspberry Pi 5 works well enough to do many of the daily tasks that the average person needs. Various programs are still unavailable for ARM Linux, so a Windows PC will remain our tool of choice for the foreseeable future.

Still, the Pi 5 makes a great second machine and is well-priced as an educational computer for children. It's also an excellent way to try out Linux if you haven't done so already. **PE**



☑ AVR development board IDE from Arduino CC
arduino-2:1.8.13+dfsg1-2 (32-bit)
☐ Command line tool for compiling Arduino sketches
arduino-builder-1.3.25-2+b5 (32-bit)
☐ Arduino Core for AVR microcontroller
arduino-core-avr-1.8.3+dfsg1-1
☐ Arduino fork of exuberant ctags
arduino-ctags-
☐ Platform files f
arduino-mighty
☐ Program your A
arduino-mk-1.5
☐ Identify, read, w … chips
flashrom-1.2-5
☐ interact with Bosch Sensortec BMP180/BMP280/BME280/BMP388 sensors (library)
golang-github-d2r2-go-bsbmp-dev-0.0~git20190515.3b4b3ae-2
☐ Debian Jr. programming

Installing packages - please wait...
Downloading packages

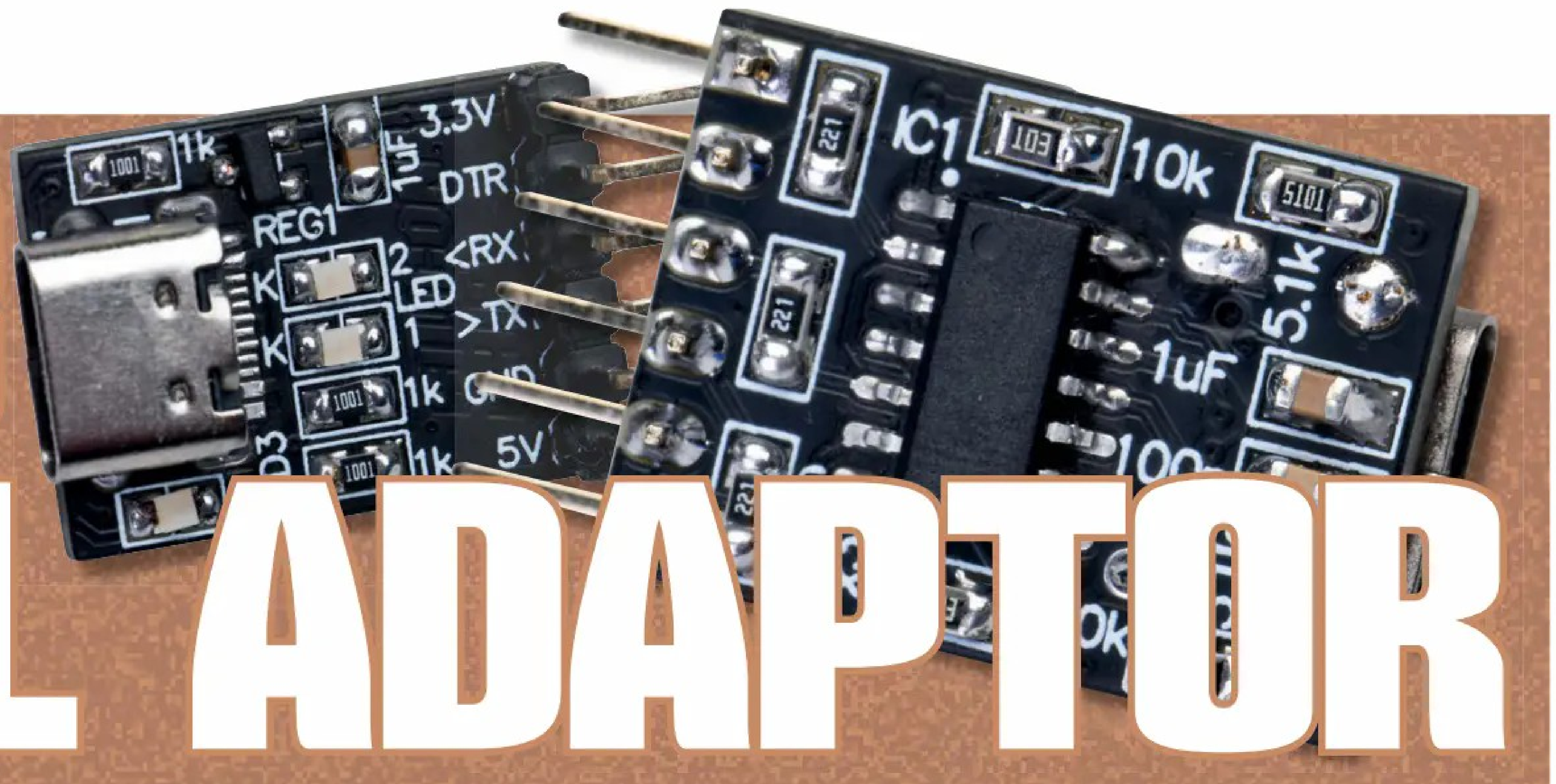
Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.
This package will install the integrated development environment that

Download size 6.1 MB
License unknown
Source debian-oldstable-main

Screen 3: Raspberry Pi Desktop is a version of the Raspberry Pi OS for x86 and x64 computers. It is a good way to try out the Raspberry Pi environment, although the hardware support is not as good as on the Pi boards (or your average PC Linux distribution). The latest version of Raspberry Pi Desktop is also a couple of years old now.

Project by Tim Blythman

USB-C SERIAL ADAPTOR



USB Type-C (USB-C) was introduced around 10 years ago and is now becoming standard. While USB-serial adaptors with Type-C sockets are available, many do not adhere to the USB-C standard and may also have Windows driver problems. Our design, presented here, has no such drawbacks and is relatively simple and compact.

We have started adding USB-C sockets to our projects as the necessary components have become available in a format that is easy to solder. Because almost all new smartphones and tablets have USB-C sockets, USB-C chargers and cables are becoming commonplace.

Small electronic modules have been a great boon for many reasons. In parallel with the rise of Arduino, they have made it very easy to connect microcontrollers to other electronic components.

We have a bit of a love/hate relationship with USB-serial adaptors. While they are incredibly useful and inexpensive, sometimes the chips used in them are clones.

You might not have any idea of that until a Windows update causes the device to stop working. A clone chip can look identical to the real deal; sometimes, the only way to tell is to X-ray it!

It isn't just a single chip that suffers from this problem. Chips labelled FT232, PL2303 and CH340G have caused problems in the past. Others may be vulnerable too.

Our design doesn't have this problem because we use a PIC microcontroller programmed to act as a USB/serial bridge, and it identifies as a generic CDC device, so there should be no way that the drivers can go wrong. Windows, Linux and macOS recognise it without needing any special drivers installed and should work immediately after being plugged in.

We have used USB-serial adaptor modules based on the CP2102 chip in several projects. One advantage of the CP2102 is that, like our design, it doesn't require drivers to work with modern operating systems.

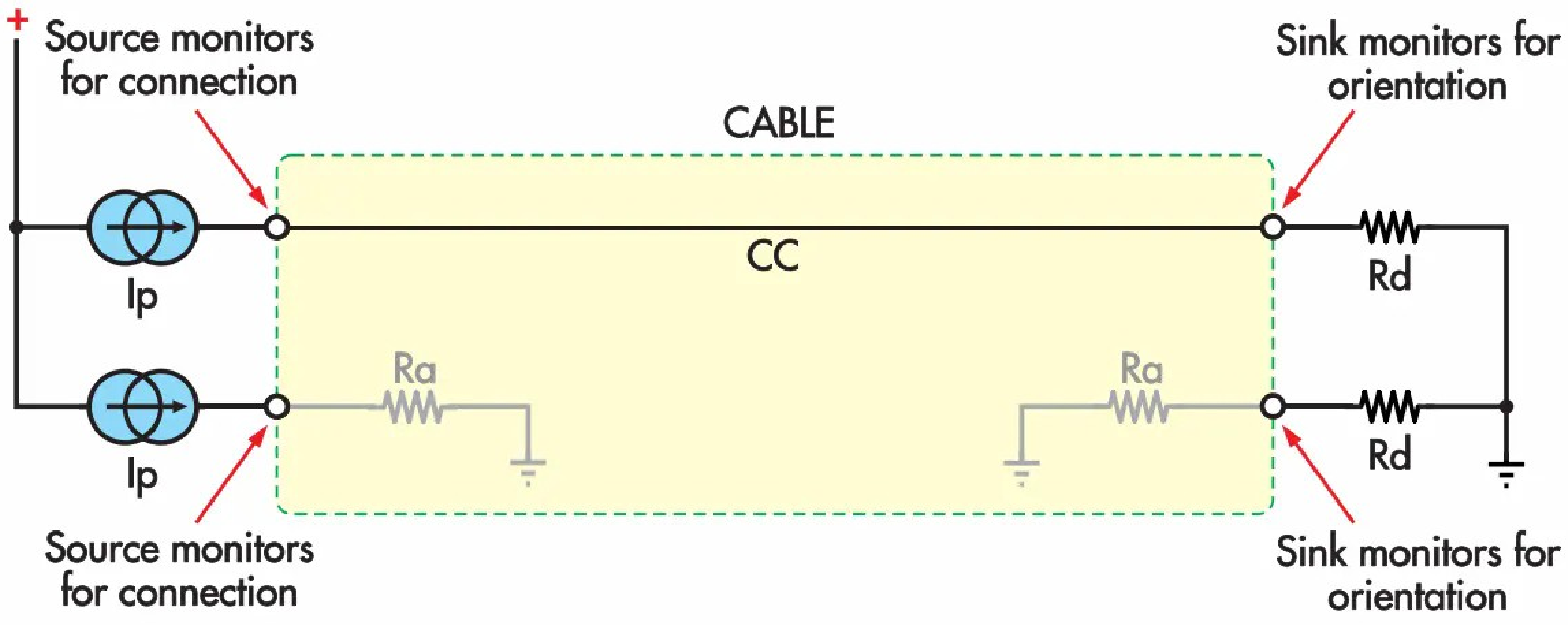

+
Source monitors for connection
CABLE
CC
Ip
Ra
Ra
Rd
Rd
Ip
Source monitors for connection
Sink monitors for orientation
Sink monitors for orientation

Fig.1: a USB-C source provides pullup currents, while a sink has pulldown resistors. Both can monitor the voltage on the CC line to determine what has connected to the other end of the cable. The source applies different currents (I_P) depending on its capacity to supply current to VBUS, which the sink can detect as differing voltages on the CC line. Advanced modes, like power delivery (PD) and dual role (DRP), are negotiated through digital signalling on the CC lines.

The most common CP2102 module is a compact device with a micro-USB socket to connect to a computer and a six-pin header to provide 3.3V logic level UART (universal asynchronous receiver transmitter, ie, serial) signals. So we have patterned our designs on that one.

USB-C advantages and challenges

USB-C is becoming ubiquitous; even Apple products like the iPhone, which have long had proprietary connectors, have switched to using USB-C, starting with the iPhone 15 in 2023.

The latest version of the Microchip PICkit debugger and programmer, the PICkit 5, also has a USB-C socket. We think that is an improvement over the micro-USB socket on its predecessor, the PICkit 4.

Although only slightly larger, in our experience, USB-C plugs and sockets are more robust than the micro-USB and mini-USB parts that preceded them. USB-C plugs and sockets are also symmetrical, which means they are less fussy to use.

USB-C to USB-C cables also exist, in which case the cable ends are even interchangeable. They are certainly less bulky than the USB sockets and plugs that appeared over 20 years ago. So it is no surprise that USB-C is becoming popular.

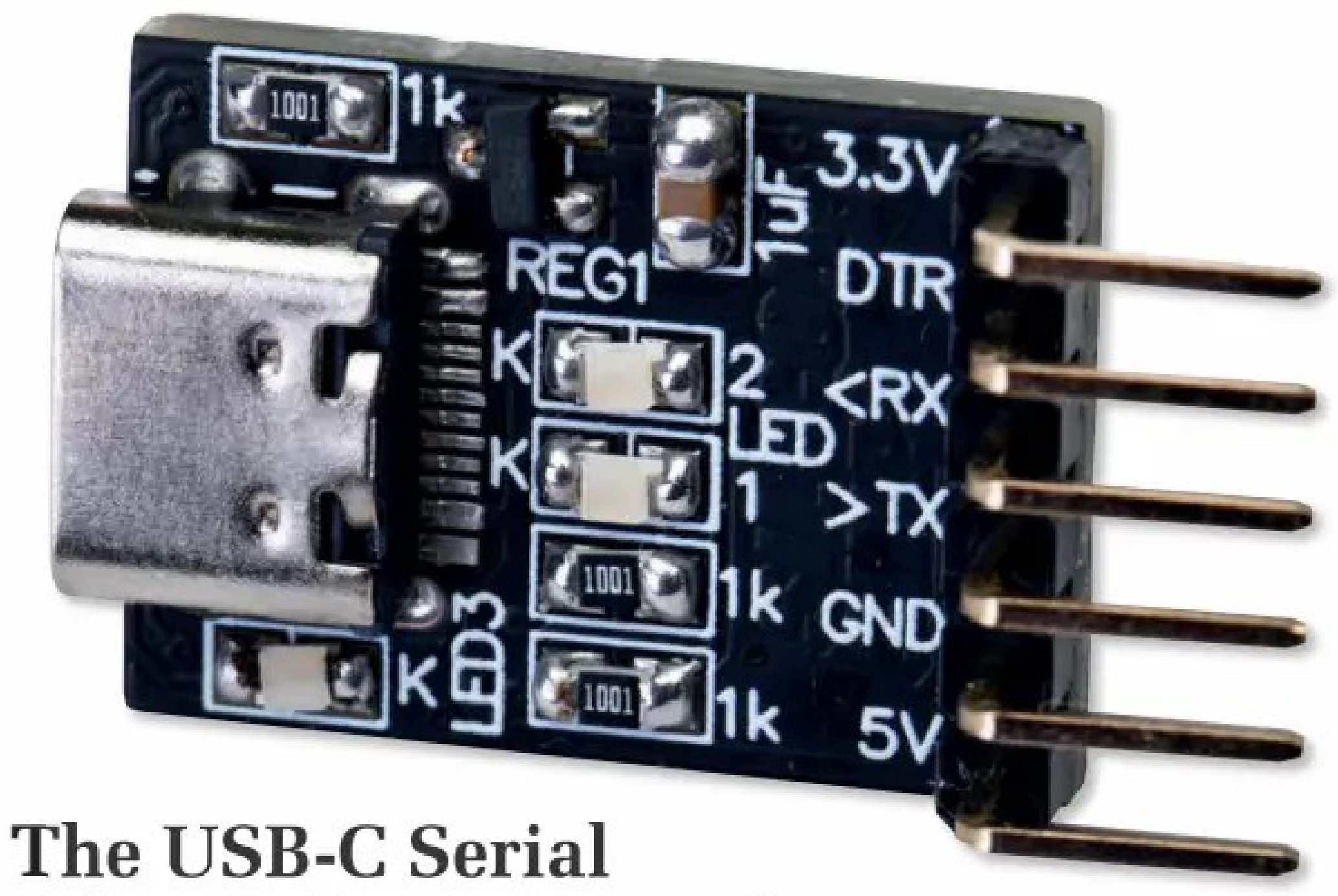
USB-C is also more complex than its predecessors and requires some knowledge to implement correctly. That has tripped up some engineers

USB-C Serial Adaptor

USB-C Serial Adaptor Features & Specifications

- Drop-in replacement for compact CP2102-based USB-serial modules with the same connector pinout
- Uses the now standard USB-C socket instead of a micro-B USB socket
- Uses a low-cost PIC16F1455 microcontroller with a USB full-speed peripheral
- Moderate component size for hand construction
- Supports 8N1 format and a wide range of baud rates (47 baud to 3Mbaud)
- 3.3V, DTR, RX, TX, GND and 5V connections
- LED indicators for power, data reception and data transmission
- No concerns about Windows drivers refusing to work with it due to counterfeit blocking attempts

USB-C Serial Adaptor Kits (SC6652, $20)
Includes the PCB, programmed microcontroller and all other parts to build the module; see the parts list later in this article.



The USB-C Serial Adaptor is a minuscule 16×22mm and operates as a drop-in replacement for the well-known CP2102 USB-serial Module. Its USB-C socket is more robust and modern than the micro-USB socket on typical USB-serial modules. The components are mostly M2012 (0805) size, but still can be hand-soldered. The USB-C socket is the finest-pitch part, so check its soldering thoroughly before applying power to the board.

who don't understand the requirements fully.

Even the Raspberry Pi Foundation had trouble with this, as their first release of the Raspberry Pi 4 had a hardware bug that meant it would not work with some USB-C cables, specifically 'smart' e-marked (with embedded electronics) cables. Older, simpler legacy cables appeared to be immune.

In simple terms, the signalling resistors used to determine the orientation and role of the cable (in combination with the CC wire in the cable) were not connected correctly. This meant that very early versions of the Raspberry Pi 4 boards were identified as audio adaptors instead of devices requesting a 5V power source and thus did not work.

Legacy cables, such as USB-A to USB-C types, lack the CC wire in the cable and thus do not respond to the incorrect signalling and deliver power regardless. Fig.1 shows how the signalling should work. There is more background on this at https://pemag.au/link/abu0

We've seen some versions of the CP2102 USB-serial modules that have replaced the micro-USB socket with a USB-C socket but they completely omitted the signalling resistors. That means that these modules will not work in all cases.

Such devices may appear to have intermittent faults, working with some cables or hosts but not others. At worst, they might not work at all as they may not receive any power.

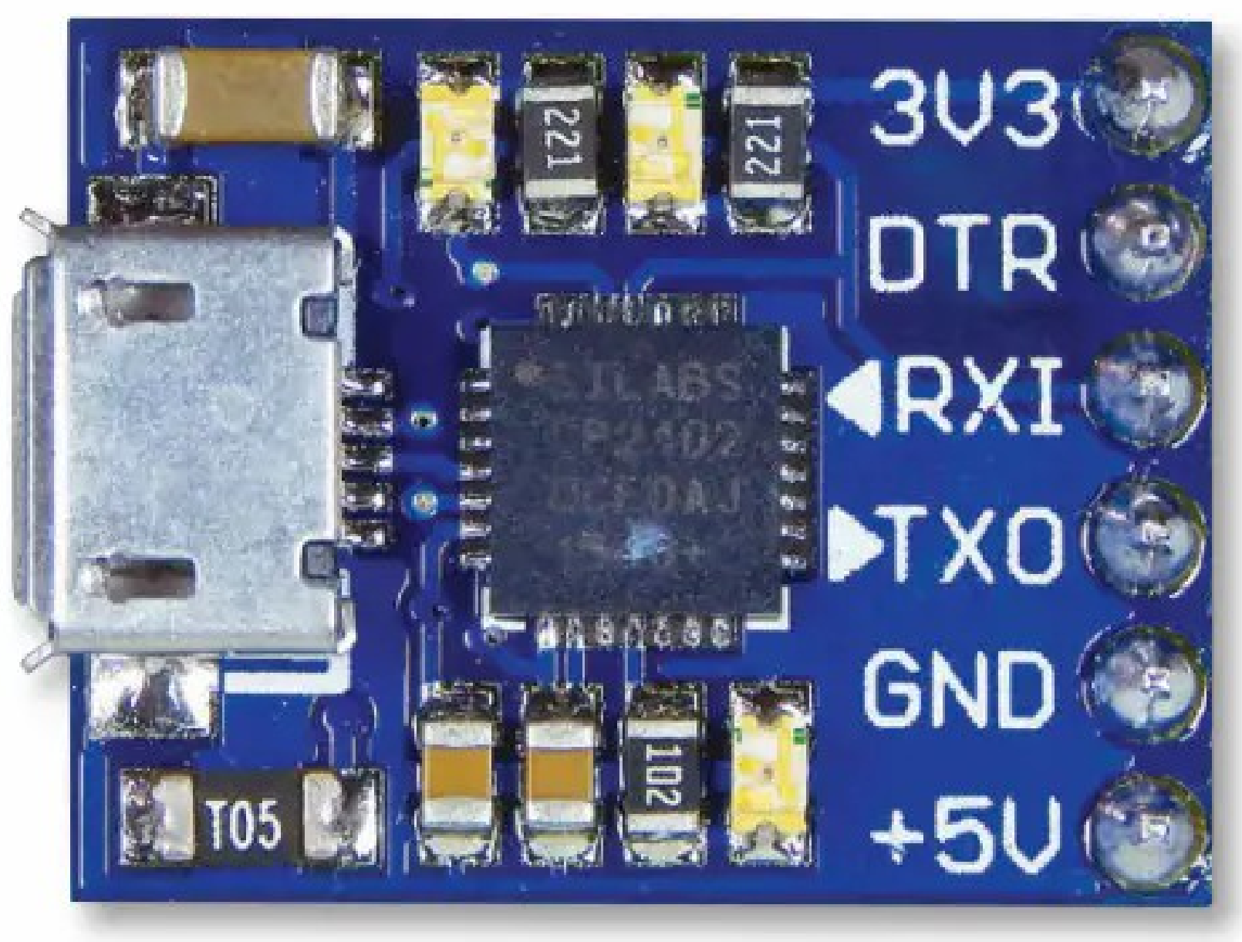
Our USB-C Serial Adaptor

So, this USB-C Serial Adaptor is a drop-in substitute for the cheap but functional CP2102 USB-serial Module and it actually works reliably!

Our Adaptor is a small PCB with a USB-C socket at one end and a six-way header at the other. Unlike the prebuilt modules you can buy, this is a constructional project you must assemble yourself. We have used some small parts, but it should be eminently doable for those with much experience in SMD soldering.

It uses a PIC16F1455 microcontroller for its USB interface. The PIC16F145x family is one of the cheapest programmable chips with a USB peripheral. We've used the PIC16F1455 in several projects, most notably the Microbridge from the May 2018 issue.

The Microbridge provides a similar USB-serial function as our Adaptor but can also program PIC32 chips. However, the Microbridge doesn't break out the DTR (data terminal ready) signal like the CP2102 module. The Microbridge also has a different connector pinout, meaning it is not a drop-in replacement for the Module.



One of the many types of CP2102-based modules, which our USB-C Serial Adaptor is meant to replace.

Circuit details

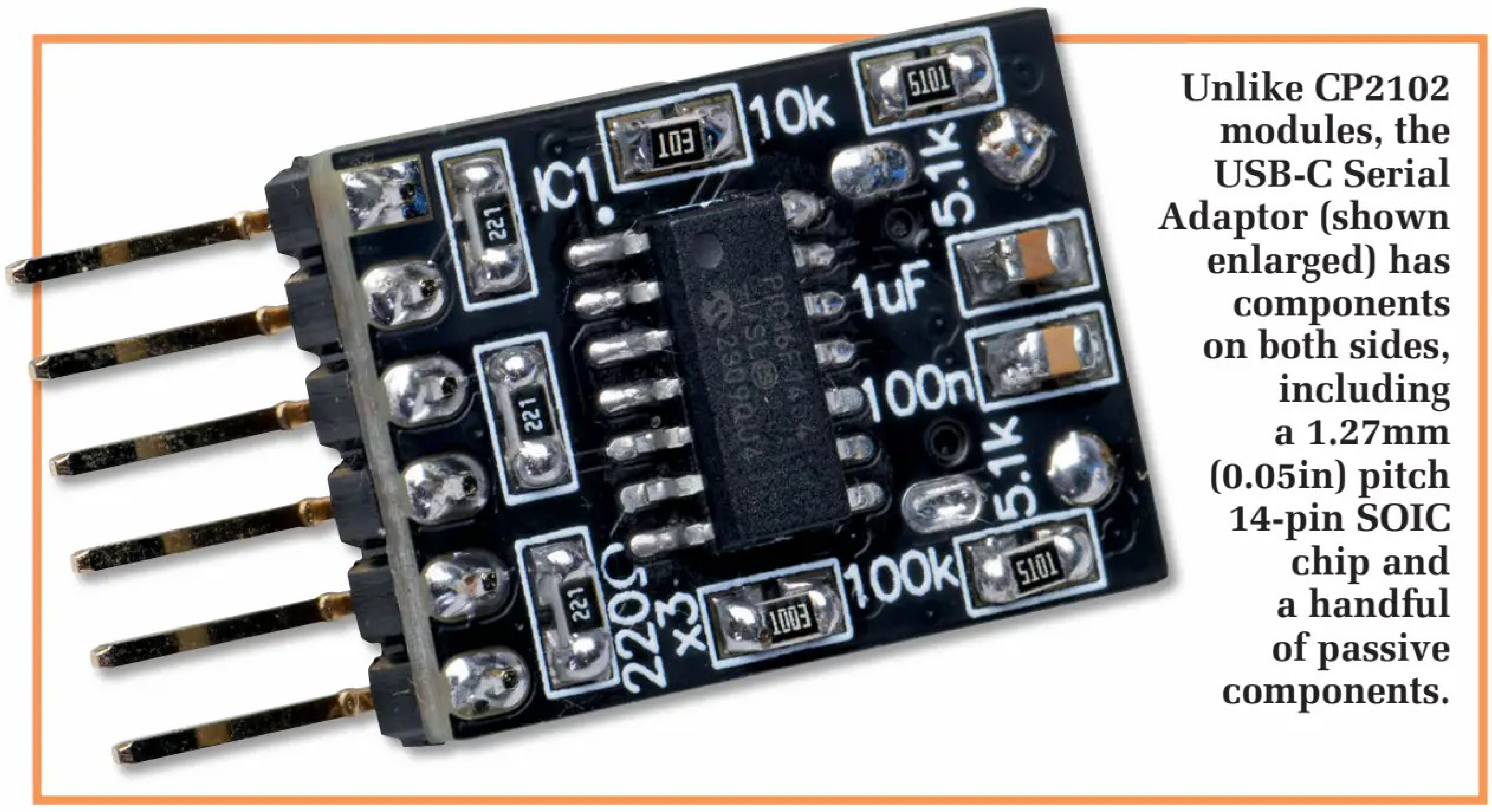
Fig.2 shows the circuit diagram of our new Adaptor. The USB socket, CON1, is a USB-C type that lacks the high-speed pairs. That means it only has one row of pins, making it easier to solder. The high-speed pairs are not needed for this design.

We previously used a USB-C socket with those extra pins in the USB Cable Tester from the November and December 2022 issues. It had two rows of very fine pins and was very fiddly to solder; the variant used in this Adaptor is easier to work with.

The SBU (sideband use) pins are present on the connector we're using, but are not needed in this design and so are not connected. The two CC pins (configuration channel) are each connected to ground via 5.1kΩ resistors, signalling that the Adaptor is a power sink (ie, it consumes power rather than provides power).

The remaining pins on CON1 are duplicated but are otherwise the same as used in standard USB 2.0 applications. The duplicated pins are simply connected together. They exist because the connector can be plugged in with two different orientations.

CON2 is a six-way pin header matching that on the CP2102 modules. It provides a means to connect to the logic-level serial signals. 5V power and ground from CON1 are connected through to CON2, as well


IC1
10k
103
5101
5.1k
1uF
221
100n
220Ω x3
100k
1003

Unlike CP2102 modules, the USB-C Serial Adaptor (shown enlarged) has components on both sides, including a 1.27mm (0.05in) pitch 14-pin SOIC chip and a handful of passive components.

as supplying REG1, an MCP1700-3.3V regulator.

It, and its two 1μF bypassing capacitors, provide the 3.3V supply to match that on the CP2102 module and so provide 3.3V logic levels. If you just wanted to get 5V and 3.3V from a USB-C cable, you could populate the Adaptor PCB with just the components mentioned so far.

PIC16F1455 microcontroller IC1 is powered at pins 1 and 14 from the 3.3V rail. There is no separate bypass capacitor because the circuit is physically very small, and the 1μF capacitor on the 3.3V rail is close to the requisite pins on IC1.

As an aside, the PIC16F1454 is much the same as the PIC16F1455, except it lacks the analog peripherals (such as the analog-to-digital converter [ADC]). We are not using any analog features, so the two chips are essentially interchangeable in this role. You should have no trouble using the PIC16F1454 if you have one on hand.

Power indicator LED3 is fed from the 3.3V rail via a 1kΩ current-limiting resistor. Serial data indicators LED1 (TX) and LED2 (RX) are driven via 1kΩ resistors from pins 9 and 10 of IC1 (digital outputs RC1 and RC0), respectively. Pin 11 of IC1 is connected to a 100nF capacitor that filters the output of a regulator internal to IC1's USB peripheral.

The USB D+ and D- signal lines (IC1's pins 13 and 12) connect to the corresponding pins on USB socket CON1 to provide the USB data interface. Pins 5, 6 and 7 on IC1 are connected to CON2 via 220Ω resistors; these are the UART RX, TX and DTR signals, respectively. The 220Ω resistors protect the microcontroller by limiting the current that can flow through the pins.

The 100kΩ resistor provides a weak pullup on the RX pin, preventing noise from being seen as data if that CON2 pin is left unconnected. The PIC16F1455 lacks an internal pullup on this pin, so we must provide this externally.

Software

The USB function is heavily dependent on software. We mentioned the Microbridge earlier; the Adaptor uses the same software library to provide the virtual USB serial port functions.

The library enumerates IC1 as a CDC (communications device class) device. CDC encapsulates the features of devices like fax machines and modems that use a serial interface, so it is well suited to working as a virtual USB-serial port.

The Adaptor software also configures pins 5 and 6 of IC1 as the UART (universal asynchronous receiver/transmitter) RX (receive) and TX (transmit) pins. Unlike newer PIC chips, these functions cannot be allocated to other pins.

In theory, the Adaptor simply needs to check the current baud rate, take data at that rate from the UART RX

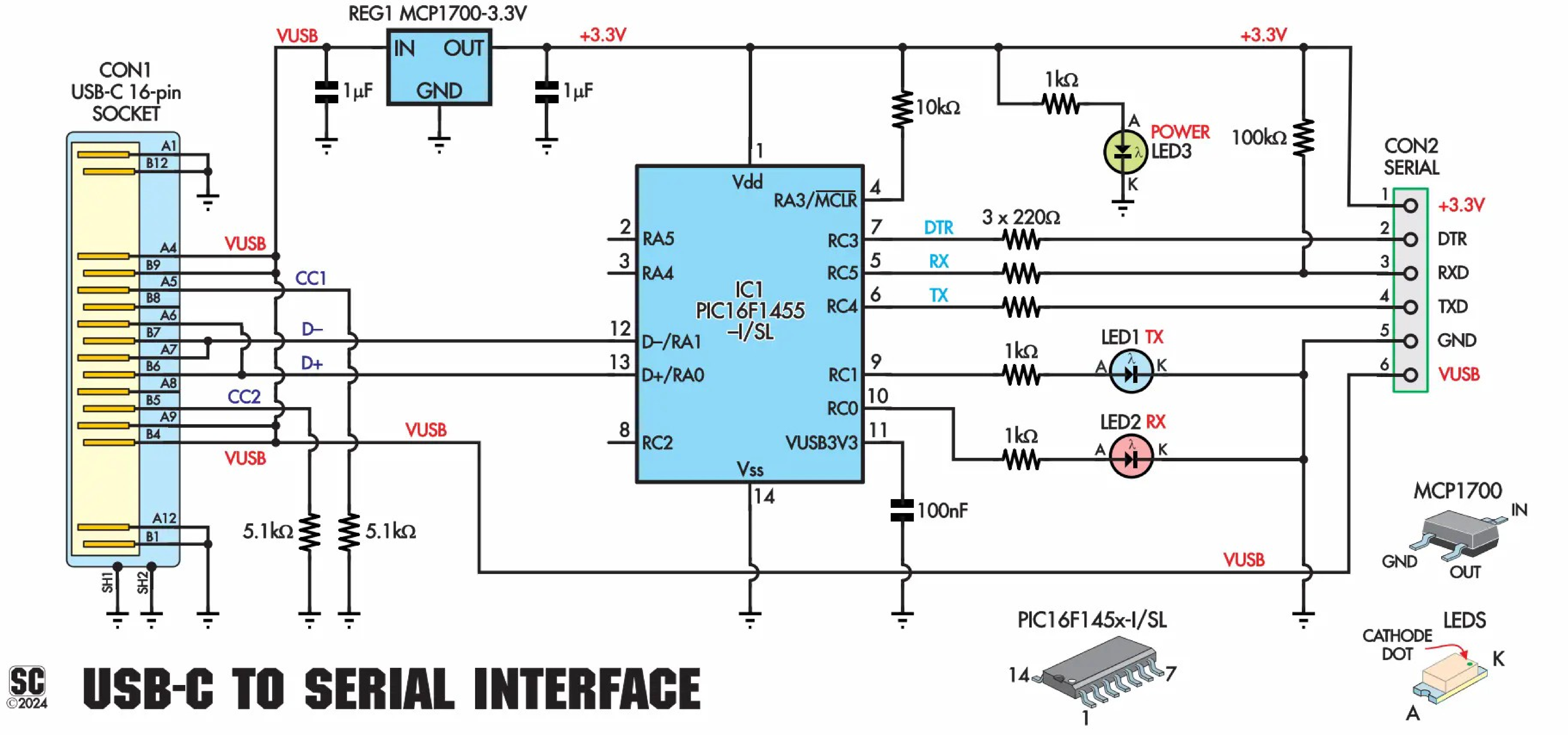

REG1 MCP1700-3.3V
VUSB
IN
OUT
GND
+3.3V
1μF
1μF
CON1
USB-C 16-pin
SOCKET
A1
B12
A4
B9
A5
B8
A6
B7
A7
B6
A8
B5
A9
B4
A12
B1
SH1
SH2
VUSB
CC1
D–
D+
CC2
VUSB
VUSB
5.1kΩ
5.1kΩ
10kΩ
1kΩ
A
POWER
LED3
K
100kΩ
+3.3V
1
Vdd
RA3/MCLR
4
2
RA5
RC3
7
DTR
3 x 220Ω
3
RA4
RC5
5
RX
IC1
PIC16F1455
–I/SL
RC4
6
TX
12
D–/RA1
13
D+/RA0
RC1
9
1kΩ
LED1 TX
A
K
RC0
10
8
RC2
VUSB3V3
11
1kΩ
LED2 RX
A
K
Vss
14
100nF
VUSB
CON2
SERIAL
1
+3.3V
2
DTR
3
RXD
4
TXD
5
GND
6
VUSB
MCP1700
IN
GND
OUT
PIC16F145x-I/SL
14
7
1
LEDS
CATHODE DOT
K
A
SC ©2024
USB-C TO SERIAL INTERFACE

Fig.2: aside from its basic functionality, the USB-C Serial Adaptor provides a few niceties, such as independently-driven TX and RX LEDs, series protection resistors for the data lines and a weak pullup on RX for noise rejection.

pin and send it to the USB host, and from the USB host to the UART TX pin. In practice, a few other things need to happen to make it compatible with the CP2102 module.

For a start, LED1 is switched on for about 50ms every time serial data is received from the USB host. Similarly, LED2 switches on whenever data is seen on the UART RX pin.

Having separate lines to drive these LEDs means that the TX and RX lines are not loaded unnecessarily. We can also show a clearer indication that data is present by lighting the LED longer than it would be if driven directly by brief pulses on the serial lines.

The DTR pin is held at a high idle level and then taken low whenever the virtual USB port is open; this means an application is actively connected to the CDC device. Also, the UART TX pin is set to a high-impedance state if a USB host is not connected. The utility of these functions may not be obvious, but they have specific uses in applications like the Arduino.

Arguably, modules like the CP2102 USB-serial adaptor exist because of the Arduino ecosystem. In early Arduino boards (before the Uno!), the DTR pin on a separate USB-serial adaptor was used to reset the microcontroller and enter a bootloader.

An RC circuit turns the high-low transition into a brief pulse for the micro's reset pin, and the bootloader runs for the first second or so after reset.

The circuit on the Uno R3 works similarly, although the USB-serial adaptor is incorporated into the board.

Allowing the TX pin to float if there is no active connection means the corresponding RX pin on whatever is attached can be used for other purposes when not needed for programming since it is not being driven.

USB data is passed in packets at times dictated by the driver in the USB host. Data is sent and received over the bus at 12Mbps (USB full-speed) during these periods. If transmission and reception are both occurring, this data must be interleaved over the bus.

Each direction has a 256-byte buffer to smooth the transition between the packetised USB data and the continuous UART data. The UART peripheral can also buffer a byte or two of data before it gets moved to or from the main buffers.

The software also monitors for packets requesting changes in the baud rate or to send a 'break' signal. A break is simply a condition where the TX line is held low for a time longer than one byte (the PIC16F1455 does this for 13 bit times). It is often used to synchronise transmission with the receiving device.

When a request for a break signal is sent from the computer, the TX LED flashes for half a second.

Limitations

We have chosen the PIC16F1455 because it is inexpensive, but that is for a reason. An 8-bit microcontroller does not have much processing power, especially for handling the amount of data that USB can move around.

As such, the Adaptor cannot do everything that a CP2102 module can. The UART peripheral on IC1 is limited to 8-bit or 9-bit data, and it does not natively support parity bits like the CP2102 chip.

To keep things simple, we only support 8-bit mode. This helps with the throughput of the Adaptor too, as there is one less special case to handle. The current version of the software uses 92% of the 1024 bytes of available RAM, so there wouldn't be space to store the 9th bit for both 256-byte buffers even if we wanted to.

Still, it can handle all the typical use cases for a USB-serial adaptor, including very low and very high baud rates.

Baud rates

The PIC16F1455 has hardware that uses the USB host's clock to tune its 48MHz internal oscillator; the available steps result in an oscillator error of up to 0.2%.

The microcontroller can produce a wide range of baud rates, from 47 to 3,000,000 baud, from the 12MHz instruction clock. Our calculations show that the error in deriving the baud rate will be less than 0.2% for the standard rates shown in Table 1.

Thus, the total error in the requested baud rate compared to the actual baud rate will be less than 0.4% for standard rates. Any arbitrary baud rate under 1Mbaud (1,000,000 baud) will have an error of less than 4%, which should be sufficient for most applications over short distances.

Table 1 – baud rate accuracy

Baud rate	Max. error
110	0.20%
300	0.20%
600	0.20%
1200	0.20%
2400	0.20%
4800	0.20%
9600	0.20%
14,400	0.24%
19,200	0.20%
38,400	0.36%
57,600	0.36%
115,200	0.36%
230,400	0.36%
250,000	0.20%
460,800	0.36%
1,000,000	0.20%

Typical error at standard baud rates (including 0.2% due to the internal oscillator).

The throughput of a USB full-speed connection is 12 megabits per second; this will not be achieved in practice, as the USB connection is usually shared with other devices. Remember that this also includes data in both directions.

In practice, the limit is much lower, primarily due to the drivers that limit the size of the packets that can be sent. We cannot easily change this, so we are somewhat stuck with that.

So continuous transmission at higher baud rates is not possible, although we had no trouble sending and receiving bursts of data up to 3Mbaud and continuous reception up to 460,800 baud.

Most of these concerns will not affect the common uses of these modules, such as acting as a programming interface for a microcontroller or handling user input (eg, on a Micromite) at baud rates between around 4800 and 115,200.

Programming

We have omitted a microcontroller programming header to keep the USB-C Serial Adaptor much the same size as the CP2102-based modules. Thus, unless you have a preprogrammed microcontroller, you should program it before soldering it to the PCB.

If you purchase a kit from the Silicon Chip shop, IC1 will be programmed, so you won't have to program it yourself.

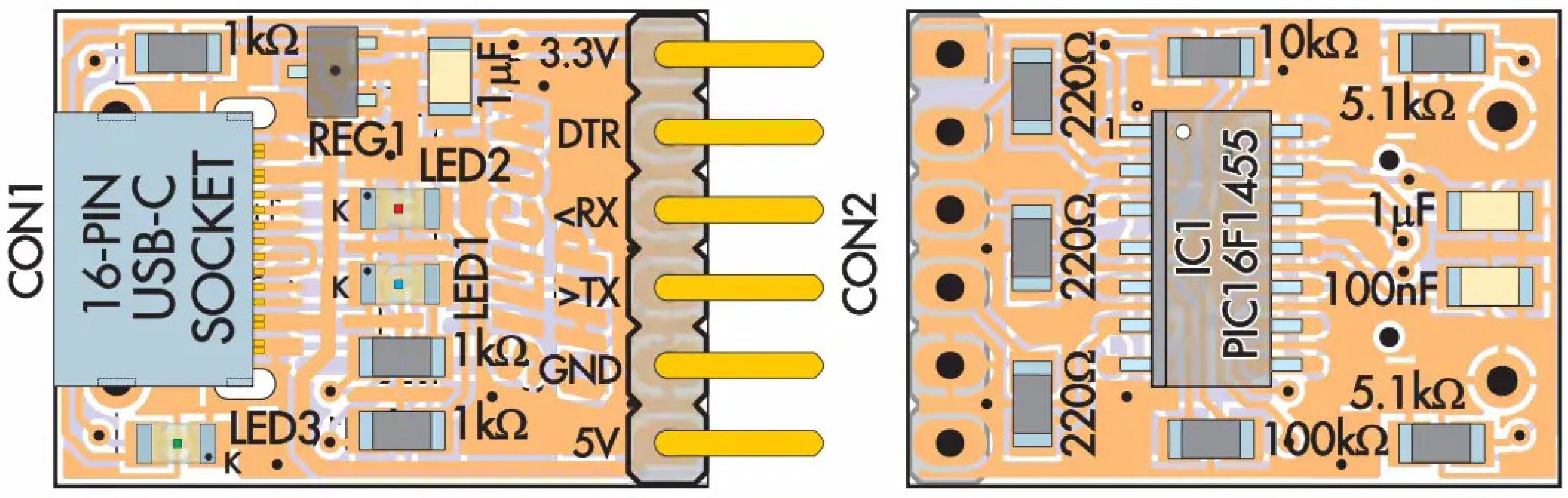
Our PIC Programming Adaptor project from the September 2024 issue has more information about the gear you might need to program an SMD chip. Note that you will also need a PICkit 3, 4 or 5 to do the programming.

To allow us to quickly reprogram our prototype during development, we soldered fine wires directly to the PIC's programming pins while it was mounted on the PCB. That is an option to consider if you only need to do this once for this project.

We used the low-voltage programming pins (pins 12 and 13) since the other programming pins (pins 9 and 10) are loaded by the LEDs, which could interfere with programming. Of course, pins 12 and 13 are the USB pins, so you should not have a programmer connected at the same time anything is connected to the USB socket.

Construction options

We've specified a right-angled header for CON2 since that is what most CP2102-based modules are supplied with. If fitting the module to a PCB, you might prefer a straight header.



TOP OF PCB (200% SCALE)

UNDERSIDE OF PCB (ALSO 200%)

Fig.3: use this diagram and the photos to ensure the many small components are all in the correct locations. Take care that IC1 is orientated correctly. If you look from the end of the chip, you should see a chamfered edge on the pin 1 side.

If you are adding the Adaptor to a low-power design, you could omit the LEDs to save on the current they would draw. In that case, you could also omit the 1kΩ resistors. The 100kΩ resistor could also be left off if you are sure that the RX pin will always be in a well-defined state.

Construction

You'll need all the standard gear for SMD work, including a good magnifier. This is one of the smaller projects we have created, and it packs the parts in fairly tightly. You might need a magnifier even to read the PCB's silkscreen markings.

Make sure you have solder flux (ideally as a paste), tweezers, a fine-tipped iron and a means of securing the board, such as Blu-Tack. Fume extraction (or working outside) will help remove flux smoke.

You should also have on hand a suitable solvent for cleaning up the PCB afterwards, and solder-wicking braid will be helpful in case a solder bridge forms.

The USB-C Serial Adaptor is built on a double-sided PCB coded

Fluke/Philips PM3092 Oscilloscope
2+2 Channel 200MHz Delay TB,
Autoset etc – £250

STEWART OF READING
17A King Street, Mortimer, near Reading, RG7 3RS
Telephone: **0118 933 1111** Fax: **0118 933 2375**
USED ELECTRONIC TEST EQUIPMENT
Check website **www.stewart-of-reading.co.uk**
(ALL PRICES PLUS CARRIAGE & VAT)
Please check availability before ordering or calling in

HP 54600B Oscilloscope
Analogue/Digital Dual Trace 100MHz
Only £75, with accessories £125

LAMBDA GENESYS	PSU GEN100-15 100V 15A Boxed As New	£400
LAMBDA GENESYS	PSU GEN50-30 50V 30A	£400
IFR 2025	Signal Generator 9kHz – 2.51GHz Opt 04/11	£900
IFR 2948B	Communication Service Monitor Opts 03/25 Avionics	POA
IFR 6843	Microwave Systems Analyser 10MHz – 20GHz	POA
R&S APN62	Syn Function Generator 1Hz – 260kHz	£295
Agilent 8712ET	RF Network Analyser 300kHz – 1300MHz	POA
HP8903A/B	Audio Analyser	£750 – £950
HP8757D	Scaler Network Analyser	POA
HP3325A	Synthesised Function Generator	£195
HP3561A	Dynamic Signal Analyser	£650
HP6032A	PSU 0-60V 0-50A 1000W	£750
HP6622A	PSU 0-20V 4A Twice or 0-50V 2A Twice	£350
HP6624A	PSU 4 Outputs	£400
HP6632B	PSU 0-20V 0-5A	£195
HP6644A	PSU 0-60V 3.5A	£400
HP6654A	PSU 0-60V 0-9A	£500
HP8341A	Synthesised Sweep Generator 10MHz – 20GHz	£2,000
HP83630A	Synthesised Sweeper 10MHz – 26.5 GHz	POA
HP83624A	Synthesised Sweeper 2 – 20GHz	POA
HP8484A	Power Sensor 0.01-18GHz 3nW-10µW	£75
HP8560E	Spectrum Analyser Synthesised 30Hz – 2.9GHz	£1,750
HP8563A	Spectrum Analyser Synthesised 9kHz – 22GHz	£2,250
HP8566B	Spectrum Analsyer 100Hz – 22GHz	£1,200
HP8662A	RF Generator 10kHz – 1280MHz	£750
Marconi 2022E	Synthesised AM/FM Signal Generator 10kHz – 1.01GHz	£325
Marconi 2024	Synthesised Signal Generator 9kHz – 2.4GHz	£800
Marconi 2030	Synthesised Signal Generator 10kHz – 1.35GHz	£750
Marconi 2023A	Signal Generator 9kHz – 1.2GHz	£700

Marconi 2305	Modulation Meter	£250
Marconi 2440	Counter 20GHz	£295
Marconi 2945/A/B	Communications Test Set Various Options	POA
Marconi 2955	Radio Communications Test Set	£595
Marconi 2955A	Radio Communications Test Set	£725
Marconi 2955B	Radio Communications Test Set	£800
Marconi 6200	Microwave Test Set	£1,500
Marconi 6200A	Microwave Test Set 10MHz – 20GHz	£1,950
Marconi 6200B	Microwave Test Set	£2,300
Marconi 6960B	Power Meter with 6910 sensor	£295
Tektronix TDS3052B	Oscilloscope 500MHz 2.5GS/s	£1,250
Tektronix TDS3032	Oscilloscope 300MHz 2.5GS/s	£995
Tektronix TDS3012	Oscilloscope 2 Channel 100MHz 1.25GS/s	£450
Tektronix 2430A	Oscilloscope Dual Trace 150MHz 100MS/s	£350
Tektronix 2465B	Oscilloscope 4 Channel 400MHz	£600
Farnell AP60/50	PSU 0-60V 0-50A 1kW Switch Mode	£300
Farnell XA35/2T	PSU 0-35V 0-2A Twice Digital	£75
Farnell AP100-90	Power Supply 100V 90A	£900
Farnell LF1	Sine/Sq Oscillator 10Hz – 1MHz	£45
Racal 1991	Counter/Timer 160MHz 9 Digit	£150
Racal 2101	Counter 20GHz LED	£295
Racal 9300	True RMS Millivoltmeter 5Hz – 20MHz etc	£45
Racal 9300B	As 9300	£75
Solartron 7150/PLUS	6½ Digit DMM True RMS IEEE	£65/£75
Solatron 1253	Gain Phase Analyser 1mHz – 20kHz	£600
Solartron SI 1255	HF Frequency Response Analyser	POA
Tasakago TM035-2	PSU 0-35V 0-2A 2 Meters	£30
Thurlby PL320QMD	PSU 0-30V 0-2A Twice	£160 – £200
Thurlby TG210	Function Generator 0.002-2MHz TTL etc Kenwood Badged	£65

HP/Agilent HP 34401A Digital
Multimeter 6½ Digit £325 – £375

HP33120A	Function Generator 100 microHz – 15MHz	£350
HP53131A	Universal Counter 3GHz Boxed unused	£600
HP53131A	Universal Counter 225MHz	£350
Audio Precision	SYS2712 Audio Analyser – in original box	POA
Datron 4708	Autocal Multifunction Standard	POA
Druck DPI 515	Pressure Calibrator/Controller	£400
Datron 1081	Autocal Standards Multimeter	POA
ENI 325LA	RF Power Amplifier 250kHz – 150MHz 25W 50dB	POA
Keithley 228	Voltage/Current Source	POA
Time 9818	DC Current & Voltage Calibrator	POA

Marconi 2955B Radio
Communications Test Set – £800

24106241 that measures 16×22mm. We'll refer to the side with the USB-C socket as the top of the PCB, with microcontroller IC1 at the bottom. The overlay diagram, Fig.3, should help you place the small components.

USB-C socket CON1 has the finest pin pitch of the parts used, so fit it first. Add a thin layer of flux to the PCB over its pads, then position the socket. There are holes to help align it, and you can add more flux to the top of the pins too.

Tack the larger end-most pins and confirm that the other pins are aligned with their pads and that the socket is flat on the PCB.

You can then solder the mounting pins to secure the location. Add flux to the tops of the mounting holes and apply the solder from below until it can be seen wicking up the pins to the top side. That way, you know this part is properly secured and won't easily be torn off the PCB.

Now solder the remaining pins of CON1. If you get a bridge between two pins, add a little more flux and use solder-wicking braid to draw it up. If you're unsure about your soldering, clean up the flux to get a better view of the pins under magnification before proceeding.

Solder REG1 in place next. This is on the same side of the PCB as CON1. Apply a little flux to the PCB pads and tack one lead, then check that the other leads are aligned before soldering them. That is the basic strategy needed for the remaining SMD parts.

This side also has the three LEDs and their 1kΩ resistors. LED1 is blue and is fitted adjacent to the TX pin on CON2, while LED2 is red and is nearer to the RX pin. LED3 is green. While it wouldn't be a tragedy if you mixed up the colours, we tried to make them easier to remember (eg, red and RX both start with the letter R).

LED1 and LED2 have their cathodes towards the USB-C socket. The cathode is usually marked with a small green dot or something similar, but it's best to check with a DMM set on diode test mode. When you touch the probes to the LED pads and it lights up, the red probe is on the anode, while the black probe is touching the cathode.

LED3 faces the opposite direction. Next, solder the 1kΩ resistors and then one of the 1µF capacitors, which should be the last SMD part on this side of the PCB.

Parts List – USB-C Serial Adaptor

1 double-sided PCB coded 24106241, 16×22mm
1 SMD USB Type-C socket with power & USB 2.0 data (CON1)
  [GCT USB4105 or equivalent]
1 6-way right-angle pin header (CON2)

Semiconductors
1 PIC16F1455-I/SL microcontroller programmed with 2410624A.HEX, SOIC-14 (IC1)
1 blue SMD LED, M2012/0805 size (LED1)
1 red SMD LED, M2012/0805 size (LED2)
1 green SMD LED, M2012/0805 size (LED3)
1 MCP1700-3302 3.3V low-dropout linear regulator, SOT-23 (REG1)

Capacitors *(all M2012/0805 X7R, 10V or higher)*
2 1µF
1 100nF

Resistors *(all SMD M2012/0805 size, ⅛W, 1%)*
1 100kΩ
1 10kΩ
2 5.1kΩ
3 1kΩ
3 220Ω

Next, flip the PCB over and fit IC1. The technique is much the same, although its pins are smaller than those on the resistors and more closely spaced (although more widely spaced than the USB socket). Make sure you put it in the right way around, with pin 1 orientated as shown!

Apply flux to the PCB, place the chip with tweezers and tack one lead. Check its alignment, then solder the other leads.

It is best to fit the other 1µF capacitor next so that it doesn't get mixed up with the 100nF capacitor that mounts next to it. The other seven parts are an assortment of resistors; ensure the correct values go in the right places, as shown in Fig.3.

Now use a solvent to clean off any flux residue, allow the board to dry, then inspect it closely for bridges or dry solder joints. If everything looks good, you can solder your choice of CON2 and proceed with testing.

Testing

Try connecting the Adaptor to a USB supply. If you are not confident, don't connect it to a computer, but use a USB power supply or something similar. You should see green LED3 illuminate within a second or so. If it does not, disconnect the Module and recheck the component placement and soldering.

You could try flipping the USB-C cable to see if it makes any difference. If it does, that points to a problem with CON1 or the two 5.1kΩ resistors.

While it is plugged into a power source, use a voltmeter to measure the 3.3V and 5V pins on CON2 relative to GND. A lack of 5V indicates a problem with CON1 or the 5.1kΩ resistors. If 5V is present but 3.3V is not, there could be a problem with the regulator, or perhaps another component is shorting the 3.3V rail.

Once everything is working, connect the Adaptor to a computer and check that a new serial port is available. Use a program like TeraTerm or minicom to open the port and send some data by typing in the terminal window. You should see blue LED1 (next to TX) flash.

If you connect the RX and TX pins on CON2 (eg, using a jumper cable) and send data, the red and blue LEDs should flash together as data is being looped back. Your terminal should echo the characters you are typing.

If this is all as expected, the USB-C Serial Adaptor is working and can be deployed to your project.

Using it

The USB-C Serial Adaptor is generally a drop-in replacement for the CP2102 modules that it is intended to succeed. Like those modules, we use it to power and connect to projects for debugging purposes.

We have also incorporated such modules into projects before. You can use the USB-C Serial Adaptor instead of the micro-USB Type-B version specified in that project.

Our Adaptor has some components on the bottom side, unlike the CP2102 modules, so it will need to be spaced away a little from the host PCB. The plastic insulation on standard pin headers should be sufficient for that purpose.

MAX'S COOL BEANS

By Max the Magnificent

WEIRD & WONDERFUL ARDUINO PROJECTS

Part 5: creating & optimising display algorithms

I'm a big fan of the black-and-white movies of yesteryear. Having said that, I can't imagine modern films without colour. The light-emitting diode (LED) array on the retro games console we are currently constructing already looks spectacular when displaying black (off) and white (all elements on), but it will look even more awesome when we add colour to the mix.

Each pixel ('picture element') in our array has red, green and blue (RGB) components. The brightness of each of these components is specified by an eight-bit (one byte) field, which can represent 2^8 = 256 different combinations. We use these combinations to represent values from 0 (fully off) to 255 (fully on) in decimal.

Those numbers can also be written as 0b00000000 to 0b11111111 in binary or 0x00 to 0xFF in hexadecimal.

When we combine our three 8-bit RGB elements, we get 24-bit colour depth, providing 16,777,216 possible hues, shades, tints and tones. However, in our console, we will limit ourselves to just 12 colours (besides black and white). These will be our primary colours and their corresponding secondary and tertiary counterparts.

Primary colours are any three (or more) colours that can be mixed to provide a range of hues. In our case, the three primary colours we'll use are red, green and blue. Our three secondary colours—yellow, cyan and magenta—are obtained by mixing pairs of our primary colours in **equal** amounts.

Our six tertiary colours—flush orange, chartreuse, spring green, azure, electric indigo and rose—are obtained by mixing pairs of our primary colours in **unequal** amounts (100% of one and 50% of the other), as shown in the Fig.1 colour wheel.

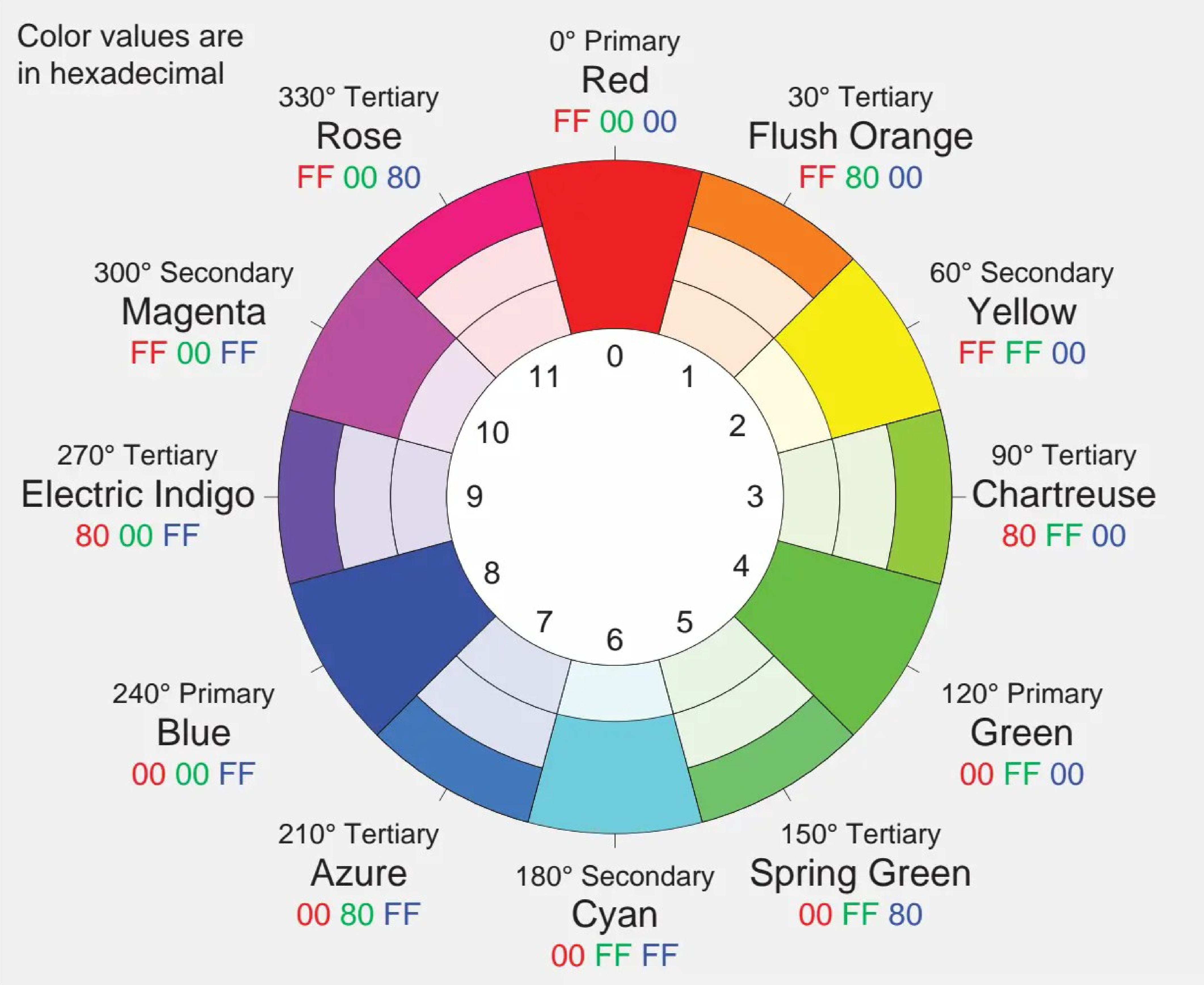

Color values are in hexadecimal
0° Primary
Red
FF 00 00
30° Tertiary
Flush Orange
FF 80 00
60° Secondary
Yellow
FF FF 00
90° Tertiary
Chartreuse
80 FF 00
120° Primary
Green
00 FF 00
150° Tertiary
Spring Green
00 FF 80
180° Secondary
Cyan
00 FF FF
210° Tertiary
Azure
00 80 FF
240° Primary
Blue
00 00 FF
270° Tertiary
Electric Indigo
80 00 FF
300° Secondary
Magenta
FF 00 FF
330° Tertiary
Rose
FF 00 80
0
1
2
3
4
5
6
7
8
9
10
11

Fig.1: a 24-bit RGB colour wheel showing some easy-to-generate colours.

It shows the colour values in hexadecimal (without the "0x" prefix), so full-off values are 00, half-on values are 80 (128 in decimal) and full-on values are FF (255 in decimal). As discussed in my previous column, to keep our power consumption at a reasonable level (and to avoid making our eyes water), we are limiting ourselves to driving our pixels at only 12.5% of their full potential.

This means that our programs will actually use values of 0x00, 0x10 and 0x20 (decimal 0, 16 and 32) for our full-off, half-on and full-on values, respectively.

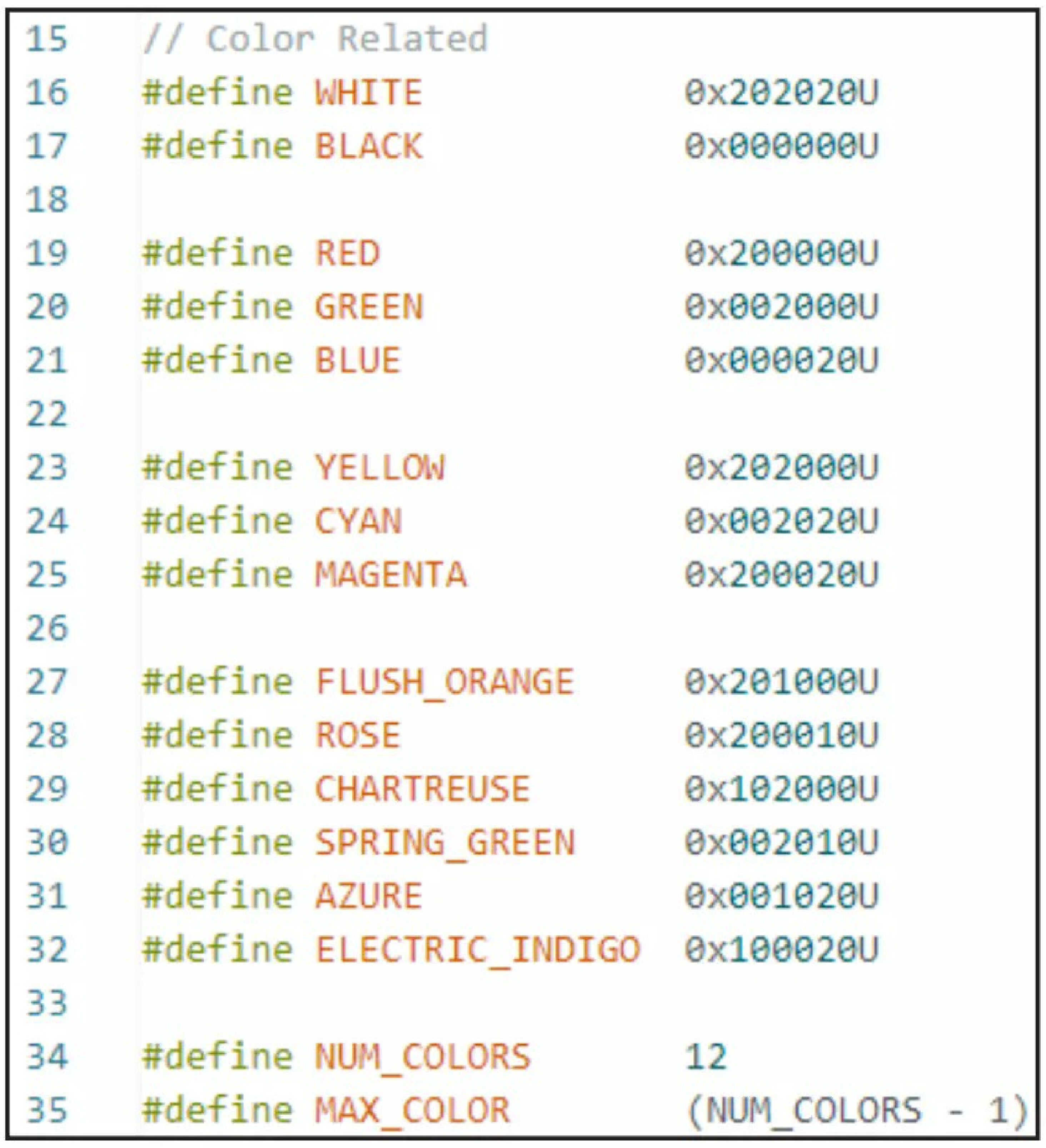
Reinventing the wheel

We will now modify one of our existing programs to display these colours on different columns. You can find a downloadable copy of this new program, named **CB-May25-Code-01.txt**, as part of the May 2025 download package from the PE website: **https://pemag.au/link/ac4y**

First, we define our colours, as shown in Listing 1(a). We are using the convention that the listing number (1 in this case) corresponds to the numerical part of the matching code file name ("01" here).

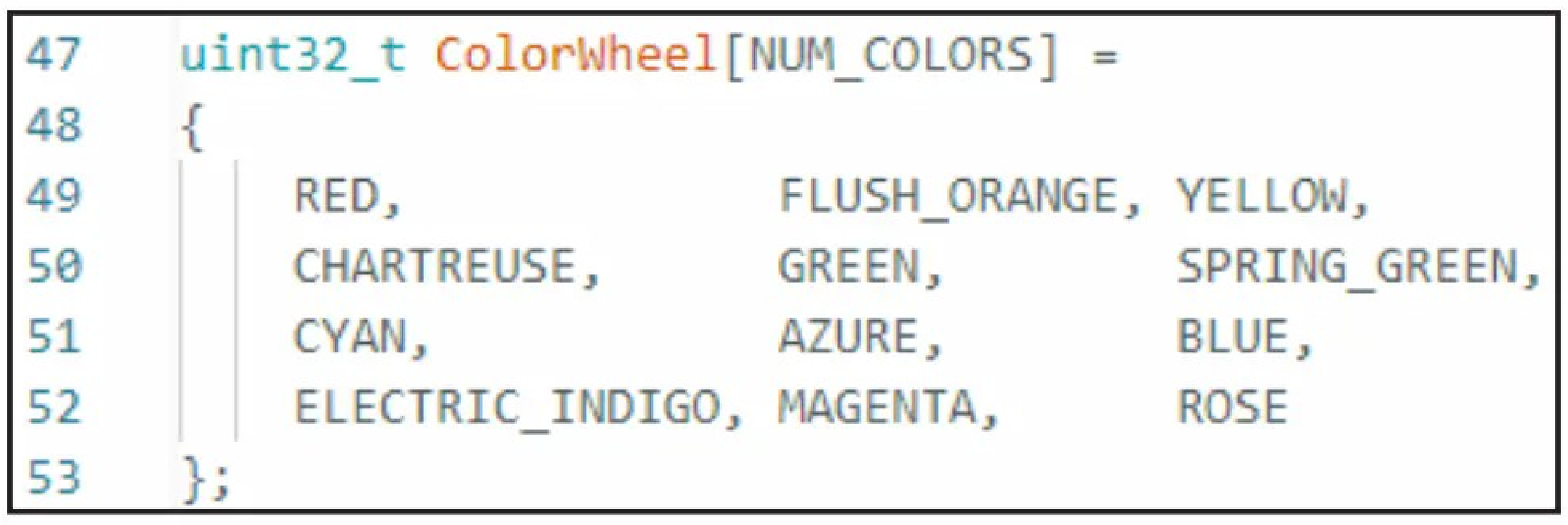
From previous discussions, we know that our Arduino compiler can only work with 8-bit, 16-bit and 32-bit integer values. However, we are using 24-bit values in our definitions. For example, we are defining **WHITE** to be **0x202020U**, where each digit represents four bits.

When we use these definitions in the program, the compiler will automatically pad the most-significant eight bits with zeroes to boost them to the required 32-bit width.



```
15  // Color Related
16  #define WHITE            0x202020U
17  #define BLACK            0x000000U
18
19  #define RED              0x200000U
20  #define GREEN            0x002000U
21  #define BLUE             0x000020U
22
23  #define YELLOW           0x202000U
24  #define CYAN             0x002020U
25  #define MAGENTA          0x200020U
26
27  #define FLUSH_ORANGE     0x201000U
28  #define ROSE             0x200010U
29  #define CHARTREUSE       0x102000U
30  #define SPRING_GREEN     0x002010U
31  #define AZURE            0x001020U
32  #define ELECTRIC_INDIGO  0x100020U
33
34  #define NUM_COLORS       12
35  #define MAX_COLOR        (NUM_COLORS - 1)
```

Listing 1(a): our colour definitions.



```
47  uint32_t ColorWheel[NUM_COLORS] =
48  {
49      RED,             FLUSH_ORANGE, YELLOW,
50      CHARTREUSE,      GREEN,        SPRING_GREEN,
51      CYAN,            AZURE,        BLUE,
52      ELECTRIC_INDIGO, MAGENTA,      ROSE
53  };
```

Listing 1(b): the colour wheel array definition.

The **U** characters at the end of our definitions are to ensure that the compiler understands we want to treat these as unsigned integer values. If we omit these **U** (or **u**) characters, 99% of the time, there won't be a problem because the compiler will figure things out correctly. However, it's always best to add these characters, nipping any strange behaviours and unexpected bugs in the bud.

NUM_COLORS (line 34) is defined as 12 because we are counting only the primary, secondary and tertiary colours (with black and white as special cases). Also observe that **MAX_COLOR** (line 35) is defined as (**NUM_COLORS - 1**). We could have defined this as 11, but this makes it easier should we decide to change the number of colours later.

Doing this incurs no overhead to our program's runtime speed (at least, in this example) because the C/C++ preprocessor will determine that the value of **MAX_COLOR** is 11 before handing things over to the compiler. To be honest, I'm not sure if we are going to need this definition (if not, we can always remove it later), but I thought I'd include it 'just in case'.

Whenever we create an equation-based definition like the one shown on line 35, it's **critical** to wrap the equation part in parentheses (round brackets), ie, '(' and ')'. Otherwise, the compiler may misinterpret our intent, which can cause our programs to misbehave.

Our next task is to declare an array of our colour values, as shown in Listing 1(b). These act as the code equivalent of our colour wheel. Numbers 0 through 11 in the centre of Fig.1 are the same as the index values in our array. So, **ColorWheel[0]** is **RED**, **ColorWheel[1]** is **FLUSH_ORANGE**, **ColorWheel[2]** is **YELLOW** etc.

Our colour wheel array will allow us to select colours algorithmically, like **ColorWheel[i]**, where the value of **i** has been determined by an algorithm (one we have yet to write).

Let there be colour!

In the **loop()** function shown in Listing 1(c), we employ the **SetColColor()** utility function we created last month. This function accepts two arguments: the number of the column we wish to change, and the colour we wish to change it to.

The **for()** loop that begins on line 67 uses the **col** variable to cycle through the columns from 0 to 11. Line 69 employs the **col** variable for two different tasks: to specify the column of interest, and to index into our **ColorWheel[]** array.

Starting with the left-hand column, lines 67 through 72 light the first 12 columns (0 to 11) with the colours from our array, with a delay of ON_TIME between each column. We leave column 12 untouched (black), then lines 75 and 76 light the right-hand column (13) with white.

On line 77, we pause for a delay of **GAP_TIME**. Then, on lines 80 through 85, we cycle through each of the columns, switch them off again. This all looks a lot cooler in the real world than you might imagine.

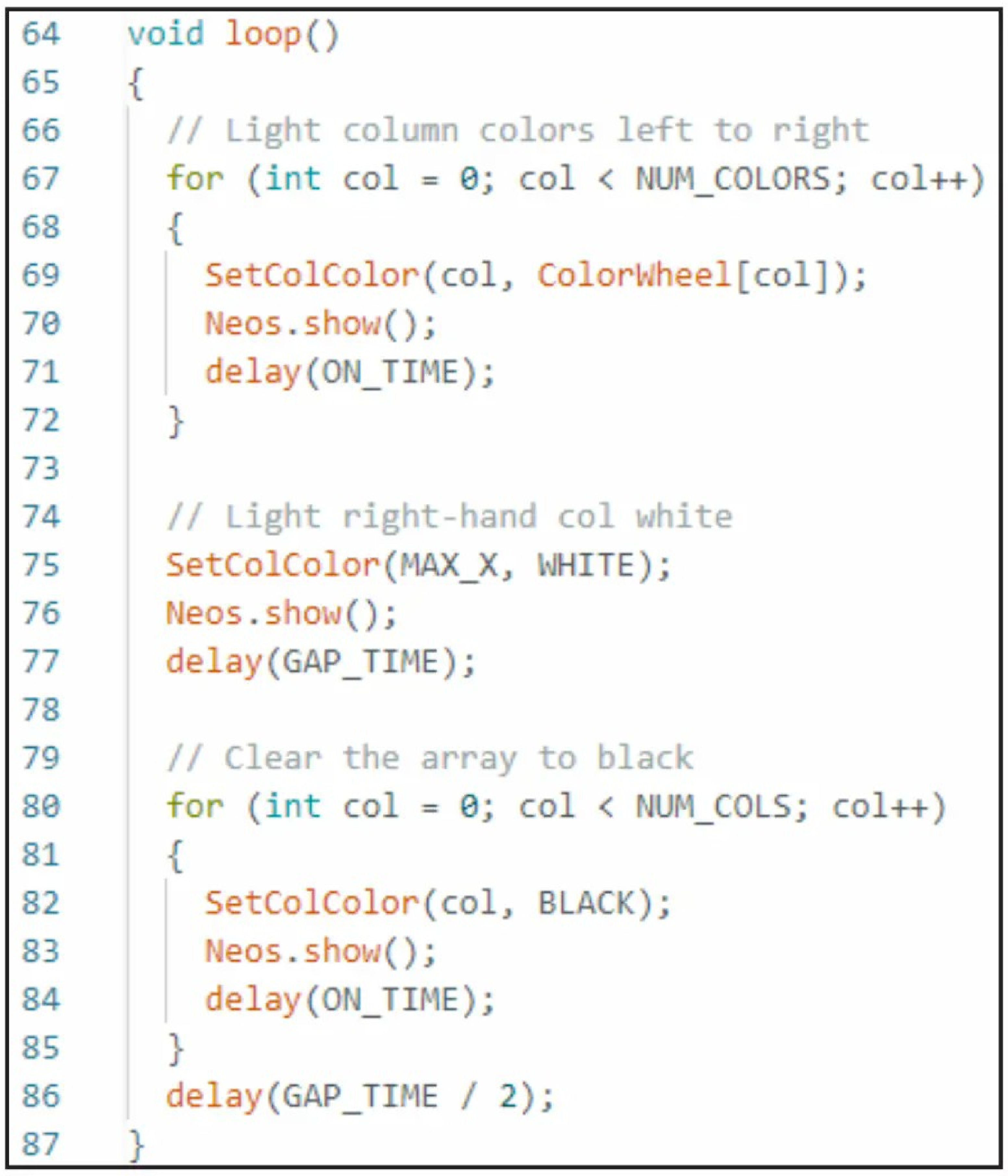
Just for giggles and grins, I created a modified version of this program to illuminate diagonal lines with different colours (see the file named **CB-May25-Code-02.txt**). From my previous column, we know that we have 23 diagonals, but we have only 12 colours, which means we end up repeating them.

The only change from our previous program is the body of our **loop()** function, as seen in Listing 2(a). This should be reasonably understandable. The **for()** loop commencing on line 67 uses the **diag** variable to cycle through our 23 diagonals (0 to 22).

On line 69, we employ the **diag** variable to specify the column of interest and to index into our **ColorWheel[]** array. We also employ the **%** (modulo) operator to restrict the index to the allowable set of 0 to 11 values.

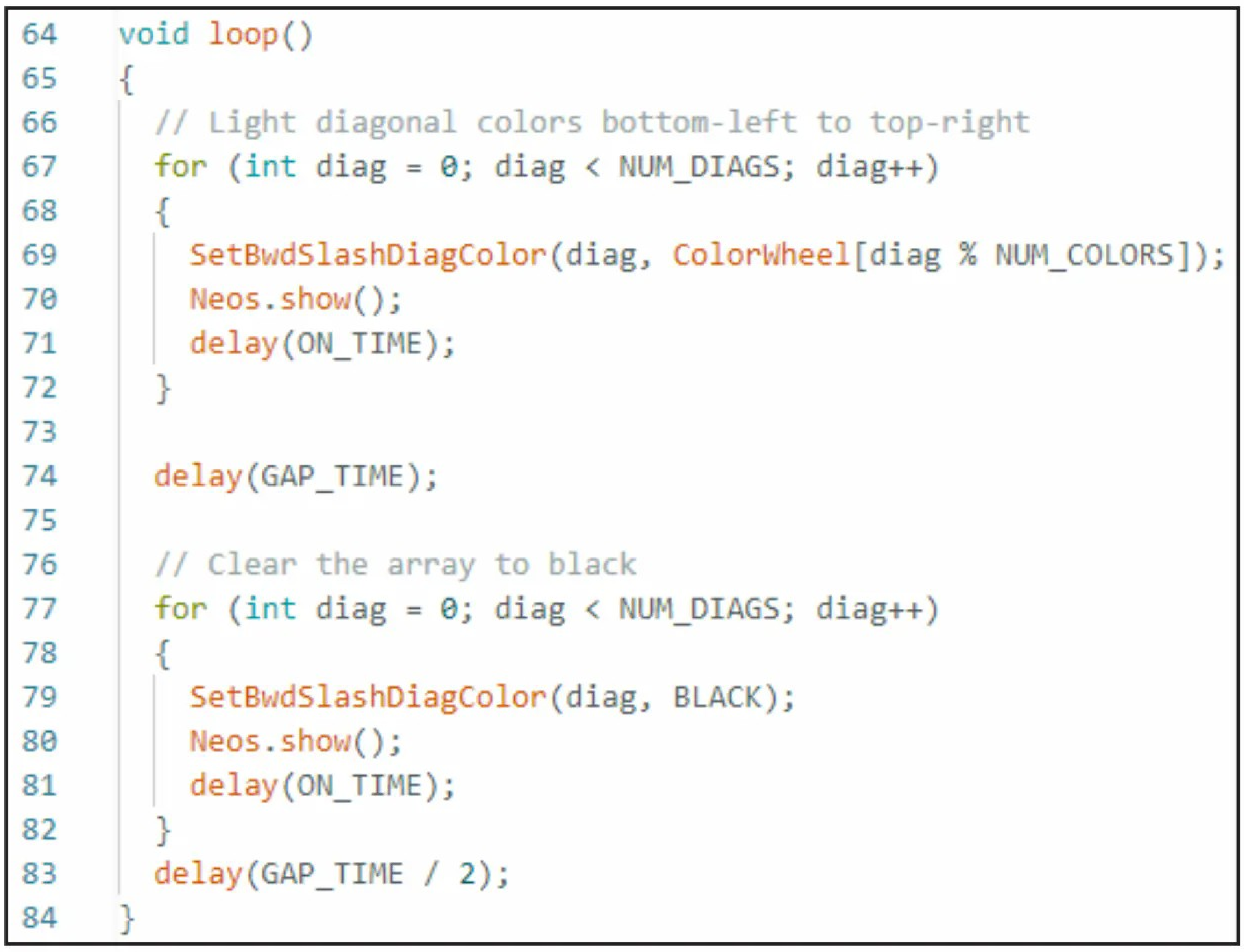
The modulo operator returns the remainder from an integer division, and we've defined **NUM_COLORS** to be 12. This means that when **diag** is 0 to 11, **diag % NUM_COLORS** will return 0 to 11. When **diag** is 12, **diag % NUM_COLORS** will return 0; When **diag** is 13, **diag % NUM_COLORS** will return 1 and so forth.

I just posted a video on YouTube of this program in action on my array (**https://youtu.be/74LTs6bOiAg**).



```
64  void loop()
65  {
66    // Light column colors left to right
67    for (int col = 0; col < NUM_COLORS; col++)
68    {
69      SetColColor(col, ColorWheel[col]);
70      Neos.show();
71      delay(ON_TIME);
72    }
73
74    // Light right-hand col white
75    SetColColor(MAX_X, WHITE);
76    Neos.show();
77    delay(GAP_TIME);
78
79    // Clear the array to black
80    for (int col = 0; col < NUM_COLS; col++)
81    {
82      SetColColor(col, BLACK);
83      Neos.show();
84      delay(ON_TIME);
85    }
86    delay(GAP_TIME / 2);
87  }
```

Listing 1(c): lighting each column with a different colour.



```
void loop()
{
  // Light diagonal colors bottom-left to top-right
  for (int diag = 0; diag < NUM_DIAGS; diag++)
  {
    SetBwdSlashDiagColor(diag, ColorWheel[diag % NUM_COLORS]);
    Neos.show();
    delay(ON_TIME);
  }

  delay(GAP_TIME);

  // Clear the array to black
  for (int diag = 0; diag < NUM_DIAGS; diag++)
  {
    SetBwdSlashDiagColor(diag, BLACK);
    Neos.show();
    delay(ON_TIME);
  }
  delay(GAP_TIME / 2);
}
```

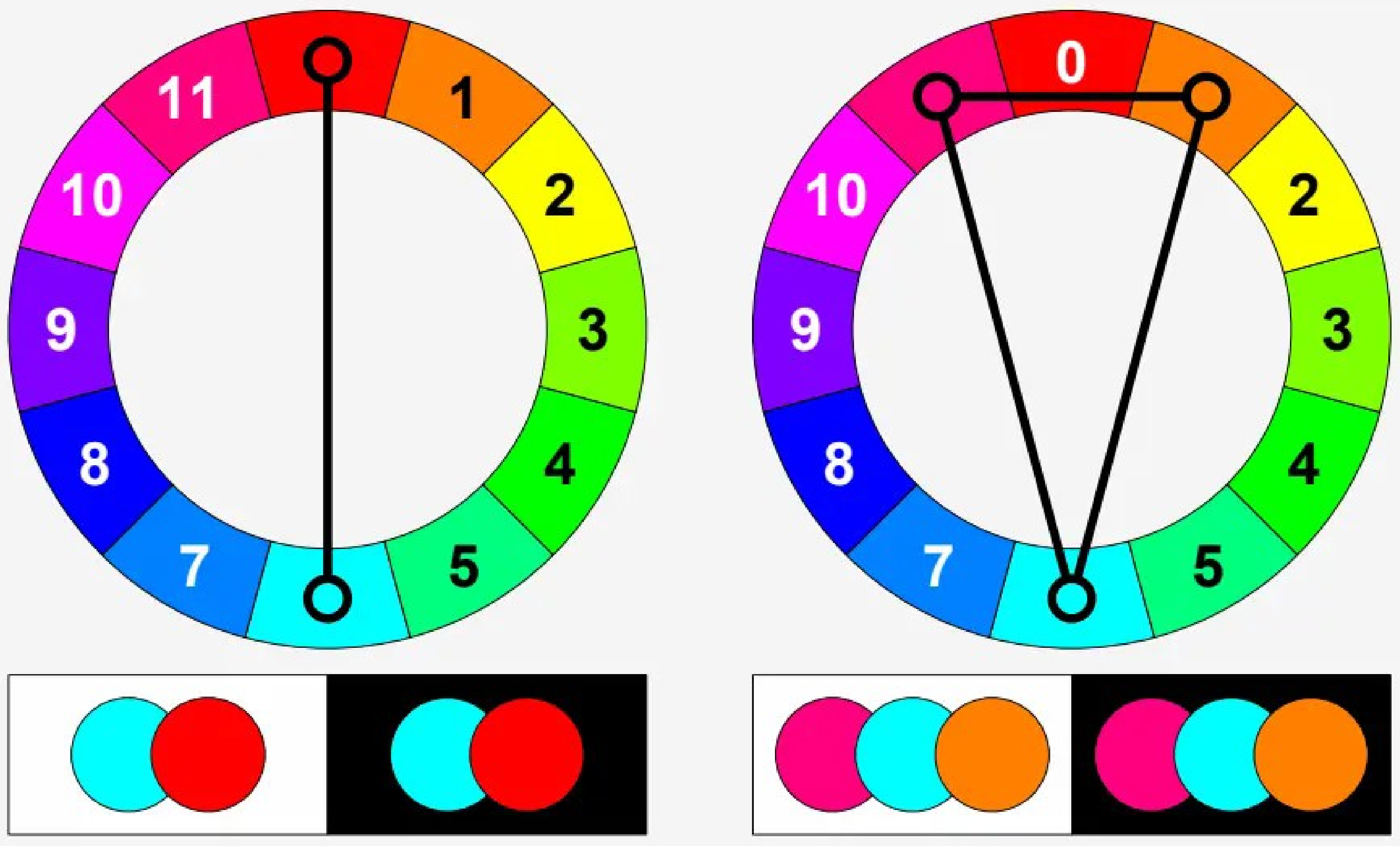
Listing 2(a): lighting diagonals with a different colour, repeating colours as required.

That's very complementary

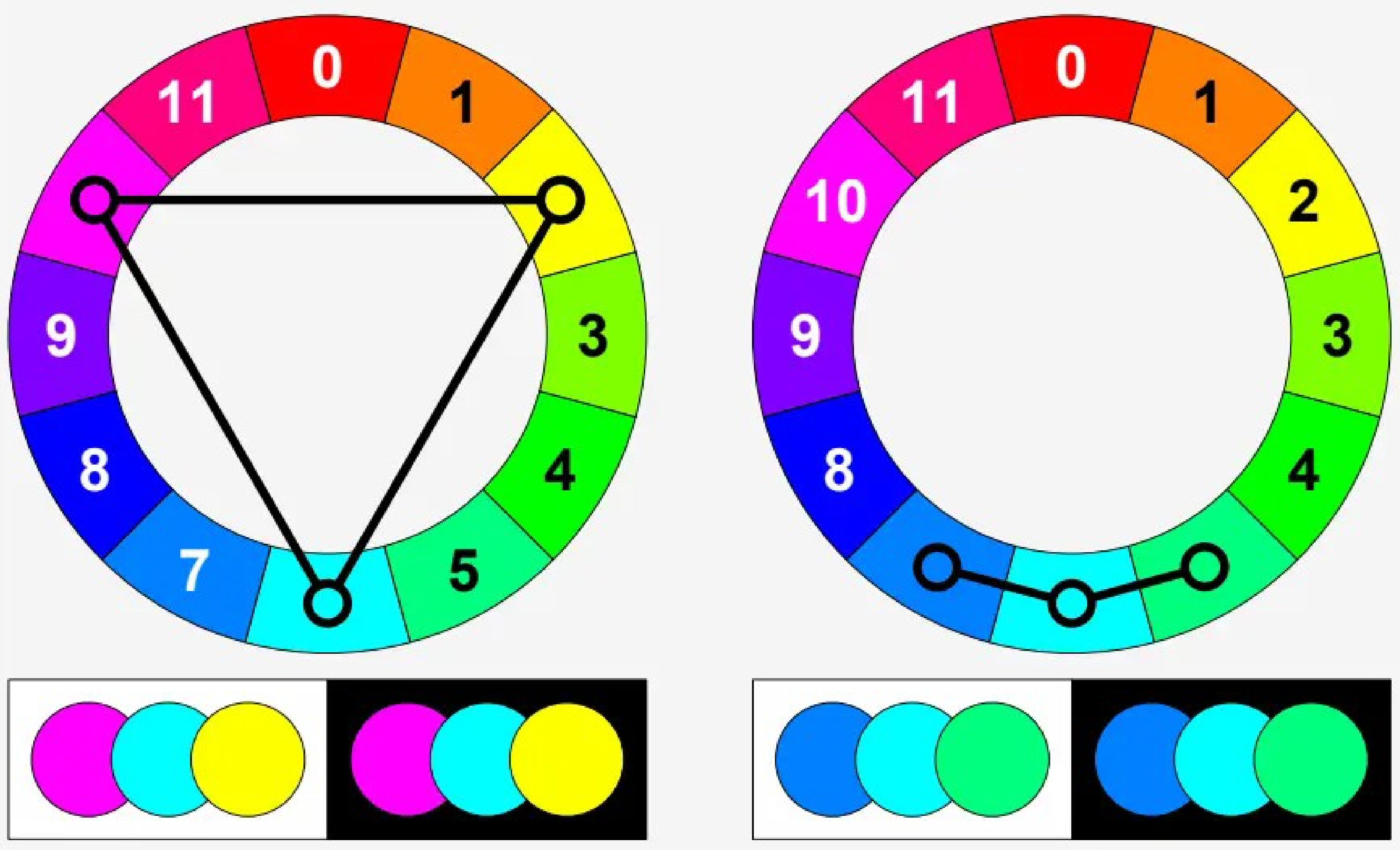
We covered colour theory in the November 2020 issue, but that was five years ago, so it's worth refreshing our minds. We may end up using this knowledge as part of our future game programs.

Every colour on our wheel has a complementary counterpart that's located 180° around the wheel. Consider the colour cyan, for example. Its complementary colour is red, and vice versa, as illustrated in Fig.2(a).

A little earlier, we noted that the main reason for creating our colour wheel array is that it allows us to select colours algorithmically. As an example, suppose we have an index value of i = 6, which equates to cyan on our wheel. We can determine the index value of its complementary colour as iCmp = (i + (NUM_COLORS / 2)) % NUM_COLORS.



(a) Complementary (b) Split Complementary



(c) Triadic (d) Analogous

Fig.2: interesting colour combinations.

Once again, we are employing the % (modulo) operator to restrict the new (calculated) index to the allowable set of 0 to 11 values. We know that NUM_COLORS is 12, meaning that NUM_COLORS / 2 = 6, i + 6 = 12 and 12 % 12 = 0.

The high contrast of complementary colours creates a lively look, but they can be jarring to the eye, which obliges us to use them judiciously.

Since we are here, let's briefly consider some additional colour combinations, starting with the concept of split complementary, illustrated in Fig.2(b). Besides the base colour (cyan in this case), we are interested in the two colours either side of the base colour's complement (rose and flush orange here).

We can determine the index values of our two split complementary colours as (i + (NUM_COLORS / 2) – 1) % NUM_COLORS and (i + (NUM_COLORS / 2) + 1) % NUM_COLORS.

We generate the index value for the complementary colour as before, then we add or subtract one to get the index values of the split complementary colours on either side.

Tricky dicky

As is usually the case, although all of this may appear simple at first, there are traps for the unwary.

The term 'edge case' refers to a scenario that occurs at an input constraint boundary. This is where the behaviour of the system might change, or unexpected problems could arise. These cases are important in software development, testing and system design because they often reveal hidden bugs.

Edge cases are different from more complicated 'corner cases', which refers to situations in which a system, algorithm, or program behaves unexpectedly or fails due to input values that are at the extreme ends (or 'corners') of those possible.

Consider the case of the analogous colours illustrated in Fig.2(d). Assuming a starting index value of i, it's tempting to think that we could generate the index values of our two analogous colours using (i – 1) and (i + 1). While this will work for mainstream i values from 1 to 10, it will fail for edge case i values of 0 and 11, which occur either side of the boundary between 0° and 360°.

With an i value of 11, (i – 1) = 10, which is what we want, but (i + 1) = 12, which is outside the bounds of our colour wheel array. Whenever one of our programs reads (or, worse, writes) outside the bounds of an array, we will not be happy with the result.

We can address this (i + 1) problem case by adding our trusty % (modulo) operator into our equations, which will give us (i – 1) % NUM_COLORS and (i + 1) % NUM_COLORS. Now, when i = 11, (i + 1) = 12 and 12 % NUM_COLORS = 0, which is what we want.

The other edge case for us to consider is when i = 0. In this case, (i + 1) = 1, which is what we want, but (i – 1) = -1, which we do not want at all. Unfortunately, our % (modulo) operator won't help us here, because (with the Arduino's C/C++) it will return a remainder of –1, which is still outside the bounds of our colour wheel array.

The solution here may appear a little unintuitive at first, but it makes perfect sense when you think about it. Before we subtract anything, we first need to travel all the way (360°) around our colour wheel in a clockwise direction by adding NUM_COLORS into the mix.

This means we will calculate the index values of our two analogous colours as (i + NUM_COLORS – 1) % NUM_COLORS and (i + NUM_COLORS + 1) % NUM_COLORS. Now, when i = 0, (i + NUM_COLORS – 1) = 0 + 12 – 1 = 11, and 11 % 12 = 11, which is what we want. Meanwhile, (i + NUM_COLORS + 1) = 0 + 12 + 1 = 13, and 13 % 12 = 1, which is also what we want.

You think this is all obvious? Good. Just to make sure, why don't you jot down the equations to generate the triadic colours as illustrated in Fig.2(c). You can find my solutions at the end of this column.

The time has come

The time has come to talk about something that is unexciting, although it is more interesting than cabbages and kings. This is another topic we covered deep in the mists of time (the September 2020 column), but it's worth mentioning again. There are various ways to declare integer variables in C/C++. We can summarise them as follows:

- signed int
- signed short int
- signed long int
- unsigned int
- unsigned short int
- unsigned long int

Signed integers can represent both positive and negative values, while unsigned can represent only positive values. As we've already discussed, an 8-bit field can represent 256 different combinations. So, an 8-bit signed integer can represent values in the range -128 to +127 (including 0), while an 8-bit unsigned integer can represent values in the range 0 to +255.

The reason the upper limits are 127 and 255, not 128 and 256, is because we need to represent zero too, so we have to sacrifice one of the non-zero numbers at the extremes to keep within our limit of 256 different values.

When we see a number like 42, we assume it to be positive. This means we don't need to write +42, although we can include the '+' if we want to. We do need to use a minus sign if the number is negative, like -42.

If we don't declare an integer as being signed or unsigned, the compiler will assume it's signed. This means that we don't need to use the signed keyword, although we can include it for clarity if we want to. We must use the unsigned keyword if we want the compiler to treat an integer as being unsigned.

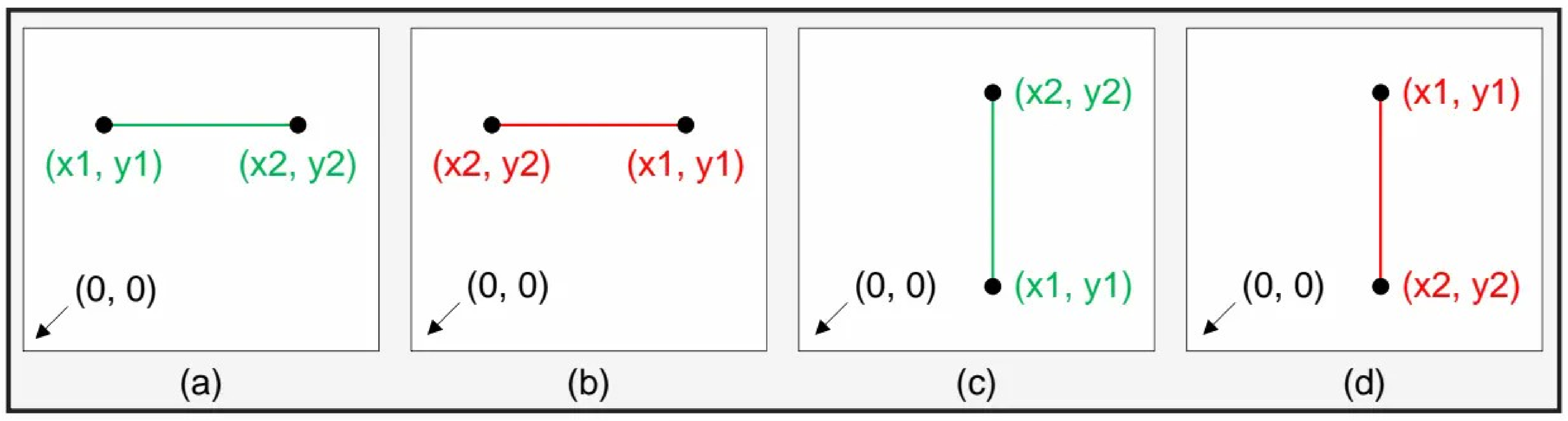


Fig.3: four ways to specify horizontal and vertical lines.

Next, there are three sizes of integer: int, short int and long int (there are larger ones, but we don't need them just yet!). If we specify short or long, we can omit the int part if we wish, because the compiler will assume we are talking about integers.

One thing that can prove tricky is that the C/C++ standards don't explicitly define the size of an int, a short, or a long. All these standards say is that an int must be a minimum of two bytes (16 bits), a short must be a minimum of two bytes (16 bits) and a long must be a minimum of four bytes (32 bits).

Apart from this, all bets are off, which can cause problems when porting a program from one computer to another.

With the Arduino Uno R3, both the int and short are 16 bits, while the long is 32 bits. The Arduino compiler also understands the type byte, which it treats as an 8-bit unsigned integer, but this typically won't be accepted or understood by other compilers.

Over time, fixed-width integer types were added to the standards. The signed types include int8_t, int16_t and int32_t, while the unsigned types include uint8_t, uint16_t and uint32_t.

Many beginners don't like the "look" of these types, preferring to use things like int, long and unsigned long. As we will soon see, however, the fixed-width types can prove extremely useful. In fact, you may remember that we used the uint32_t type to define the colour parameters in the utility functions we created in our last two columns.

Graphics utilities (lines)

Our existing SetRowColor() and SetColColor() utility functions are essentially special cases of line-drawing routines. We could replace both these functions with something a little more generic, like a DrawLine() function that can draw both horizontal and vertical lines with arbitrary lengths.

A line can be defined by two distinct (x, y) endpoints that we might call (x1, y1) and (x2, y2). Unfortunately, even though we are restraining ourselves to consider only horizontal and vertical lines, we still have four possibilities, as illustrated in Fig.3.

Remember that I'm visualising (and manipulating) my array under the assumption that its origin, (0, 0), is in the lower left-hand corner. The way I'm planning to create my function is that I will draw my lines from (x1, y1) to (x2, y2). Since I wish to keep things as simple as possible, I'm planning on incrementing my x and y values as required.

This limits me to drawing horizontal lines, as in Fig.3(a), and vertical lines, like in Fig.3(c).

What about the lines shown in Fig.3(b) and Fig.3(d)? If we were desperate, then there are various approaches we could use to address these. Let's consider the horizontal case as an example. We could start by testing to see if x1 < x2, in which case we have a Fig.3(a) situation that we can draw by **incrementing** from x1 to x2.

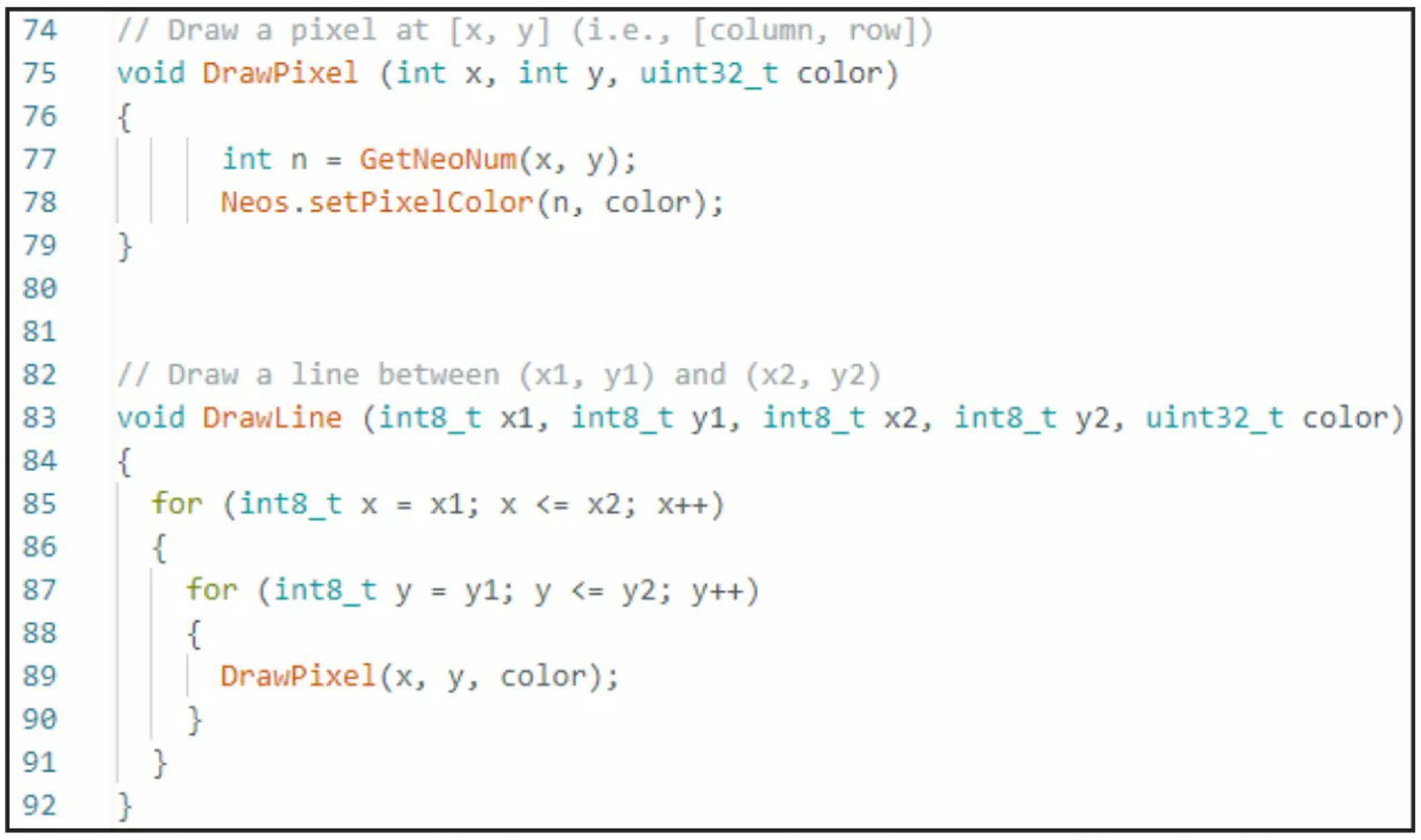
However, if x1 > x2, then we have a Fig.3(b) scenario that we can draw by **decrementing** from x1 to x2 (or we could swap our x1 and x2 values and then return to **incrementing** from x1 to x2).

The problem with doing things this way is that we are increasing the complexity of our function and the number of tasks it needs to perform, which (not surprisingly) will increase the time to perform them.

An alternative approach, which is what I'm going to do, is to say that our function will only handle horizontal lines where x1 ≤ x2 and vertical lines where y1 ≤ y2. The advantage of using ≤ rather than < is that—as you will see—our routine will draw zero-length (1-pixel) lines in which x1 = x2 and y1 = y2.

Just to make things official (and cover our posteriors), we will use a common programmer's 'get out of jail free' card, meaning that if x1 > x2 and/or y1 > y2, the output from and actions performed by this function will be undefined.

This is going to be a lot less scary than you might imagine. I just created a little test program (in the file named **CB-May25-Code-03.txt**). Our DrawLine() function appears between lines 83 and 92. If we exclude the function declaration and all the { } squiggly brackets,



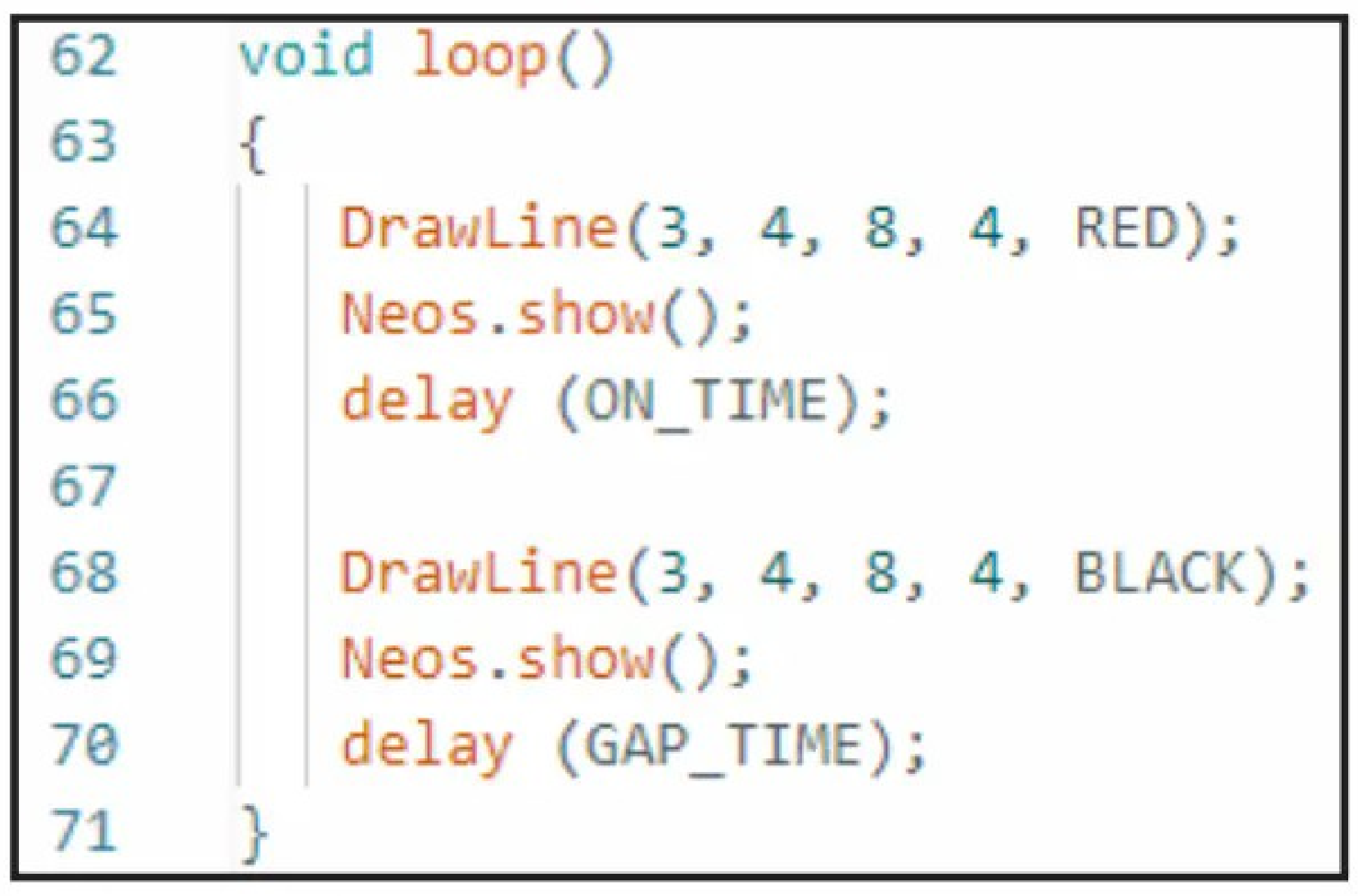
```
74  // Draw a pixel at [x, y] (i.e., [column, row])
75  void DrawPixel (int x, int y, uint32_t color)
76  {
77        int n = GetNeoNum(x, y);
78        Neos.setPixelColor(n, color);
79  }
80
81
82  // Draw a line between (x1, y1) and (x2, y2)
83  void DrawLine (int8_t x1, int8_t y1, int8_t x2, int8_t y2, uint32_t color)
84  {
85    for (int8_t x = x1; x <= x2; x++)
86    {
87      for (int8_t y = y1; y <= y2; y++)
88      {
89        DrawPixel(x, y, color);
90      }
91    }
92  }
```

Listing 3(a): our new **DrawLine()** function, along with the renamed **DrawPixel()**.

there are only three lines of code, as shown in Listing 3(a).

This function has five parameters. The first four represent the ends of the line in terms of (**x1**, **y1**) and (**x2**, **y2**) values, while the fifth is the colour we wish to use. Observe that we are making use of our new fixed-width data types (the reason for this will become obvious later in this column).

Also observe the call to the **DrawPixel()** function on line 89. It's just a new incarnation of our old **SetPixelColorXY()** function. I renamed this little scamp to make it consistent with the collection of graphics utility functions we are creating.



```
void loop()
{
  DrawLine(3, 4, 8, 4, RED);
  Neos.show();
  delay (ON_TIME);

  DrawLine(3, 4, 8, 4, BLACK);
  Neos.show();
  delay (GAP_TIME);
}
```

Listing 3(b): our first line drawing test code.

For my first test, I changed the **loop()** function to repeatedly draw and erase a red line between (3, 4) and (8, 4), as shown in Listing 3(b).

After triumphantly watching my first line flash for about an hour (I'm easily amused), I experimented with a vertical line and a few edge conditions, just to make sure everything was tickety-boo.

That all went well, so I'm happy. Still, before we proceed to the next topic, remember that I've implemented my **DrawLine()** function, assuming that (0, 0) is in the bottom-left corner of the array. If you've created your array with the origin in the top-left corner, for example, you will have to implement your **DrawLine()** function accordingly.

Graphics utilities (rectangles)

Another utility function that might come in handy is the ability to draw rectangles, which encompass squares as a subset. We can define the outer limits of a rectangle with two distinct (**x**, **y**) points that we might call (**x1**, **y1**) and (**x2**, **y2**). As for a line, there are four possibilities illustrated in Fig.4.

The same arguments we discussed for drawing lines apply to drawing rectangles. To cut a long story short (the opposite of how I like to do things), our function is going to implement only the option reflected in Fig.4(a).

Actually, we are going to create a small suite of rectangle drawing routines. We'll start with one we'll call **DrawRectOutline()**, which will draw only the outline of the rectangle. There are lots of ways we could do this. The way I chose to go is captured in a test program (in the file named **CB-May25-Code-04.txt**) and illustrated in Listing 4(a).

Like our line-drawing function, this has five parameters. The first four represent the corners of our rectangle, while the fifth is the colour we wish to use. We won't dissect this function here, except to note it places four calls to our **DrawLine()** function to draw the edges to our rectangle.

For my first test, I modified the **loop()** function to repeatedly draw and erase a red rectangle outline between (3, 3) and (9, 7), as shown in Listing 4(b).

Once I'd verified that this worked as ~~hoped~~ planned, I tweaked the corners of the rectangle to check that our algorithm handles edge conditions. These tests included extreme cases like a one-line thick rectangle, and even a single-pixel rectangle.

Random rectangles

Let's step things up a bit. We will now repeatedly draw (and then clear) rectangles with random sizes and colours. They can appear anywhere on our array. The only rule to enforce is that there must be at least one blank pixel inside the rectangle's outline. I find that sketching out a quick visual

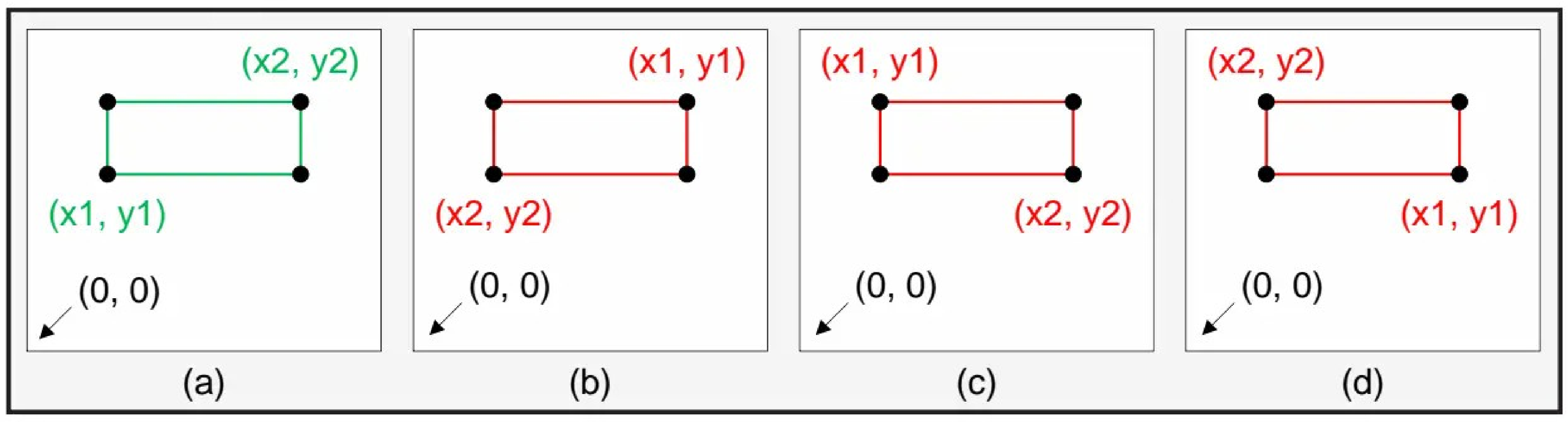


Fig.4: four rectangle specification possibilities.

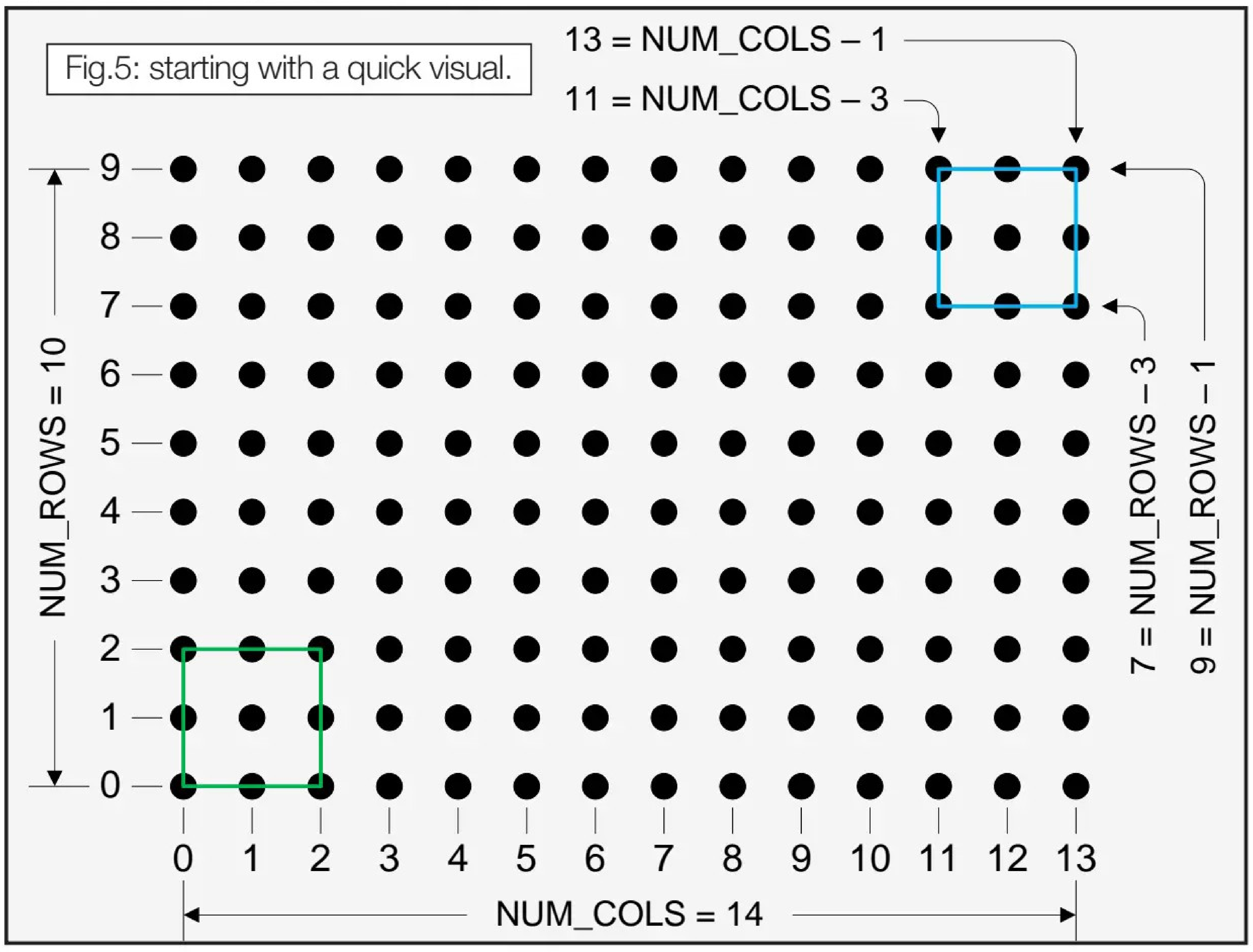
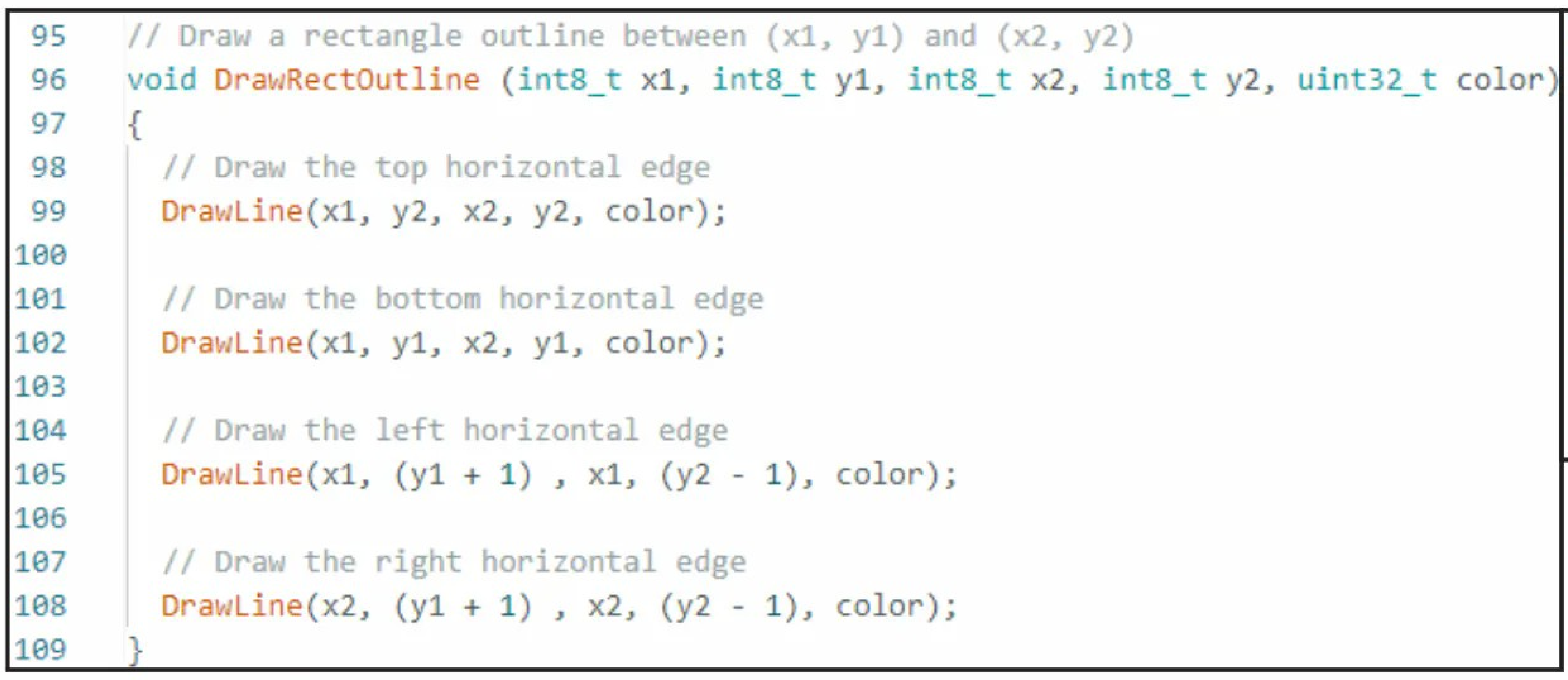


Fig.5: starting with a quick visual.



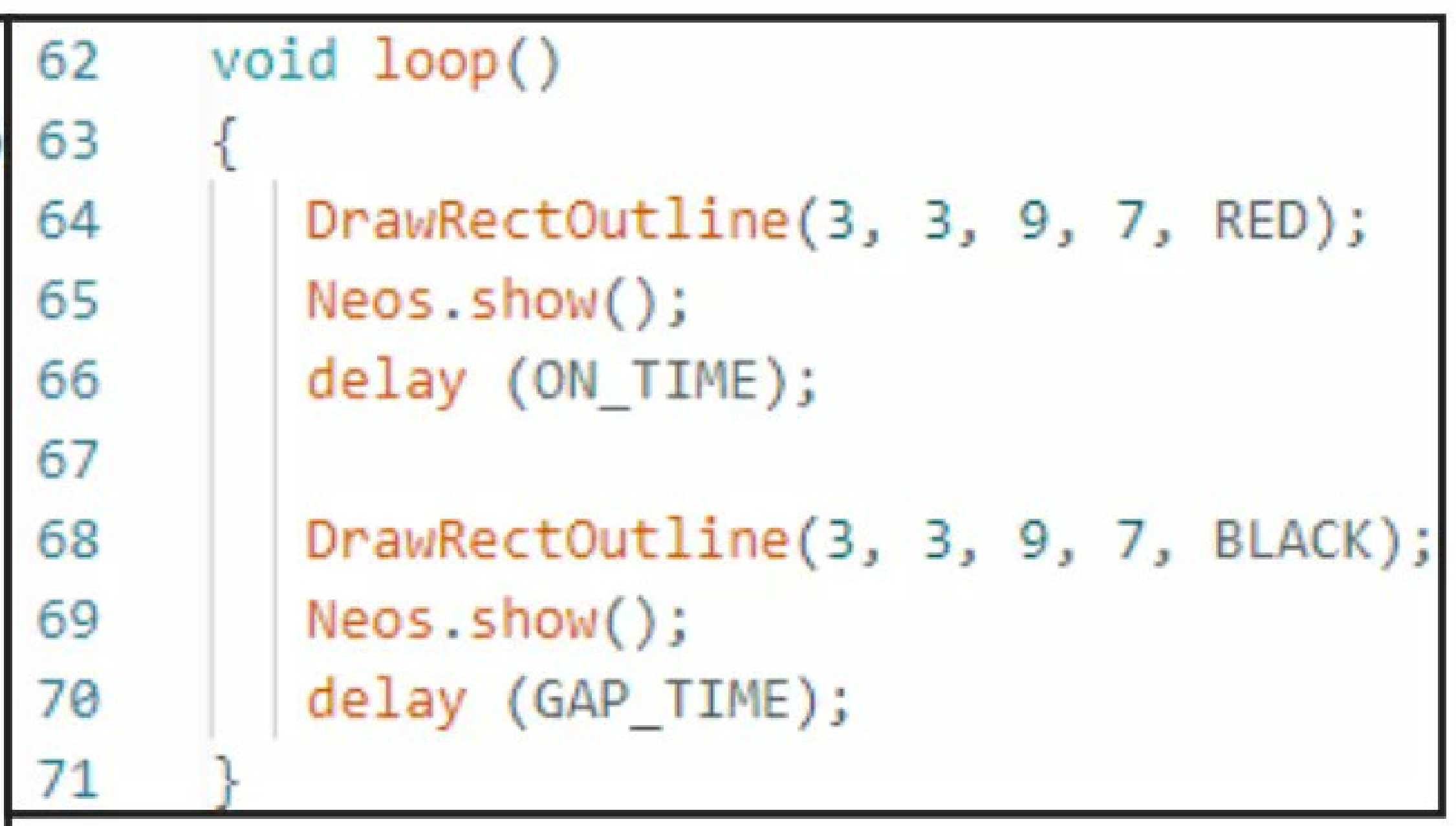
```
// Draw a rectangle outline between (x1, y1) and (x2, y2)
void DrawRectOutline (int8_t x1, int8_t y1, int8_t x2, int8_t y2, uint32_t color)
{
  // Draw the top horizontal edge
  DrawLine(x1, y2, x2, y2, color);

  // Draw the bottom horizontal edge
  DrawLine(x1, y1, x2, y1, color);

  // Draw the left horizontal edge
  DrawLine(x1, (y1 + 1) , x1, (y2 - 1), color);

  // Draw the right horizontal edge
  DrawLine(x2, (y1 + 1) , x2, (y2 - 1), color);
}
```

Listing 4(a): our DrawRectOutline() function is pretty straightforward.



```
void loop()
{
  DrawRectOutline(3, 3, 9, 7, RED);
  Neos.show();
  delay (ON_TIME);

  DrawRectOutline(3, 3, 9, 7, BLACK);
  Neos.show();
  delay (GAP_TIME);
}
```

Listing 4(b): the rectangle outline drawing test.

representation helps me wrap my brain around what I need to do (Fig.5).

We will start by generating our (x1, y1) values, followed by our (x2, y2) values. While this may seem the obvious choice, there's nothing to stop us from doing things the other way round.

As our array is 14 × 10 (width × height) pixels in size, we've defined NUM_ROWS and NUM_COLS in our programs to be 10 and 14, respectively.

The green and blue squares in Fig.5 represent a couple of worst-case (well, boundary) conditions. From these, knowing the size of our array, we can say that our x1 value can be any integer between 0 and 11. Once we've generated our x1 value, we know that our x2 value can be any integer between (x1 + 2) and 13.

Similarly, we can say that our y1 value can be any integer between 0 and 7. Once we've generated our y1 value, we know our y2 value can be any integer between (y1 + 2) and 9.

Rather than using hard-coded values in our code, we will future-proof things by basing any equations on our NUM_ROWS and NUM_COLS definitions on the off-chance we decide to change the size of our array one day.

Bearing all this in mind, I created a new version of our program with a modified version of the loop() function (in the file named **CB-May25-Code-05.txt**), shown in Listing 5(a).

We are using the Arduino's random() function to generate our random values. Based on C/C++, this function accepts two arguments that we might call min and max.

Observe line 68. Why are we using a max value of (NUM_COLS – 2) instead of the (NUM_COLS – 3) value depicted in Fig.5 (and similarly for line 69)? Similarly, on line 71, we are using a max value of NUM_COLS instead of the (NUM_COLS – 1) value shown in Fig.5.

Well, with the Arduino's random() function, the min value is **inclusive**, while the max value is **exclusive**. This means the function will generate a random integer between min and (max – 1). There is a reason for this, although it doesn't particularly suit us here.

So, a max value of NUM_COLS will be employed as the (NUM_COLS – 1) value we wish to use by the random() function. Similarly, a max value of (NUM_COLS – 2) will be treated as (NUM_COLS – 3) by the random() function.

One final point to consider is that if max is equal to (min + 1), we are essentially asking the function to generate a random value between min and min, which leaves it no option but to return a value of min.

I just posted a video on YouTube of this program running on my array (https://pemag.au/link/ac4w). I'm sorry for the poor quality of this video. Even though I'm running each active pixel at only 12.5% of its full-on brightness, that's still enough to saturate my camera. The colours look much more vibrant in real life.

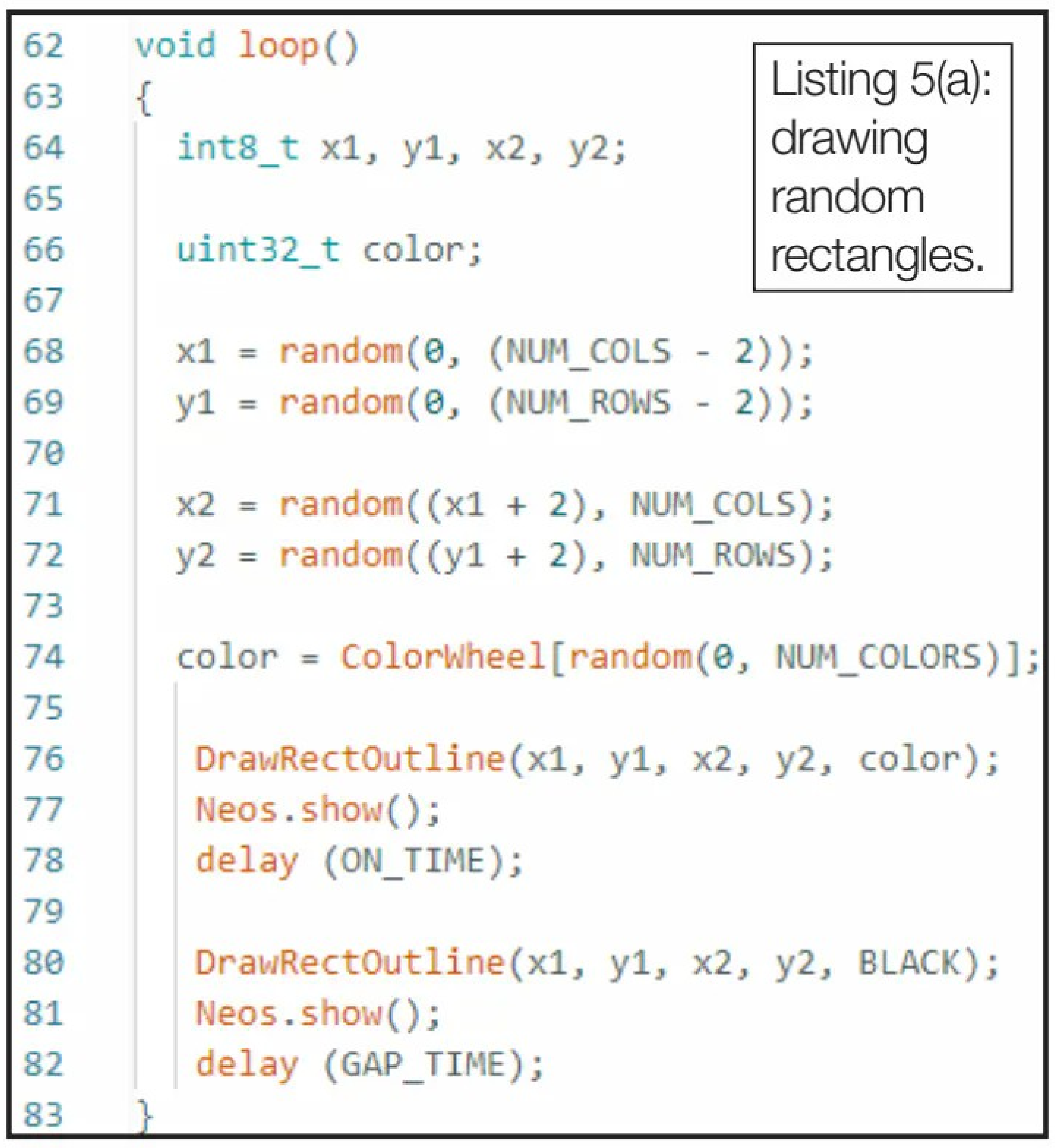
Yet more rectangles

Now let's add a DrawRectSolid() function, which will draw a rectangle filled with the same colour as its outline. This is getting easier as we go. The new version of our program containing the DrawRectSolid() function is named **CB-May25-Code-06.txt** and the new function is shown in Listing 6(a).

All we are doing is using our DrawLine() routine to draw a series of horizontal lines. In the loop() function, we exchange the two calls to the DrawRectOutline() function for calls to our new DrawRectSolid() function.

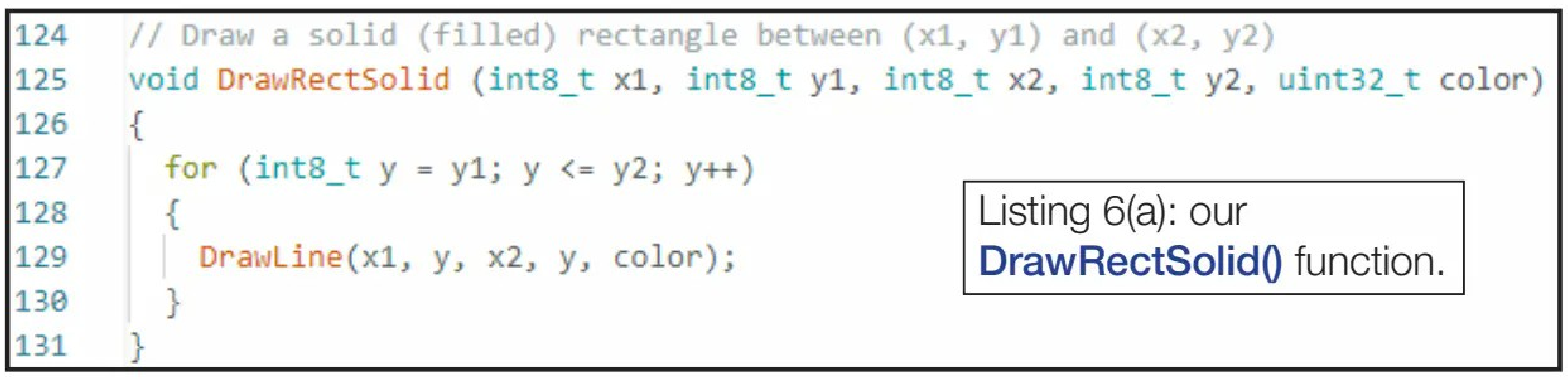
Let's also create a DrawRectFilled() function, which will draw a rectangle filled with a different colour from its outline. The code with this added is named **CB-May25-Code-07.txt** and the new function is illustrated in Listing 7(a).

We now have two colour parameters, called outlineColor and fillColor. This function makes calls to our existing DrawRectOutline() and DrawRectSolid() functions. In turn, they call our DrawLine() function, which calls our DrawPixel() function. At the conceptual



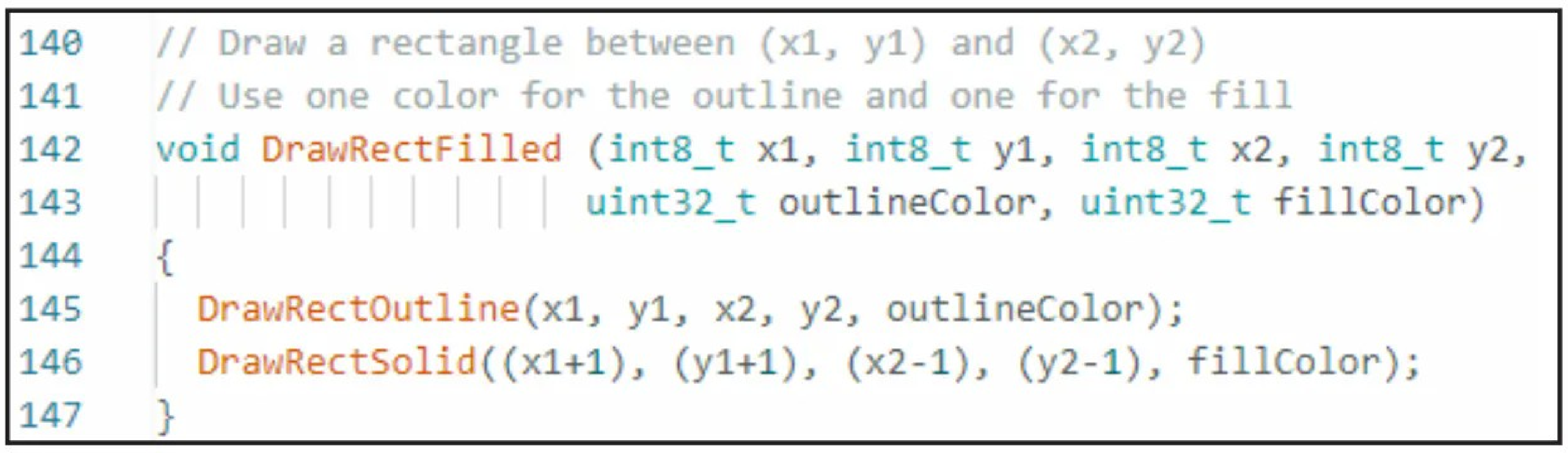
```
62  void loop()
63  {
64    int8_t x1, y1, x2, y2;
65
66    uint32_t color;
67
68    x1 = random(0, (NUM_COLS - 2));
69    y1 = random(0, (NUM_ROWS - 2));
70
71    x2 = random((x1 + 2), NUM_COLS);
72    y2 = random((y1 + 2), NUM_ROWS);
73
74    color = ColorWheel[random(0, NUM_COLORS)];
75
76     DrawRectOutline(x1, y1, x2, y2, color);
77     Neos.show();
78     delay (ON_TIME);
79
80     DrawRectOutline(x1, y1, x2, y2, BLACK);
81     Neos.show();
82     delay (GAP_TIME);
83  }
```

Listing 5(a): drawing random rectangles.



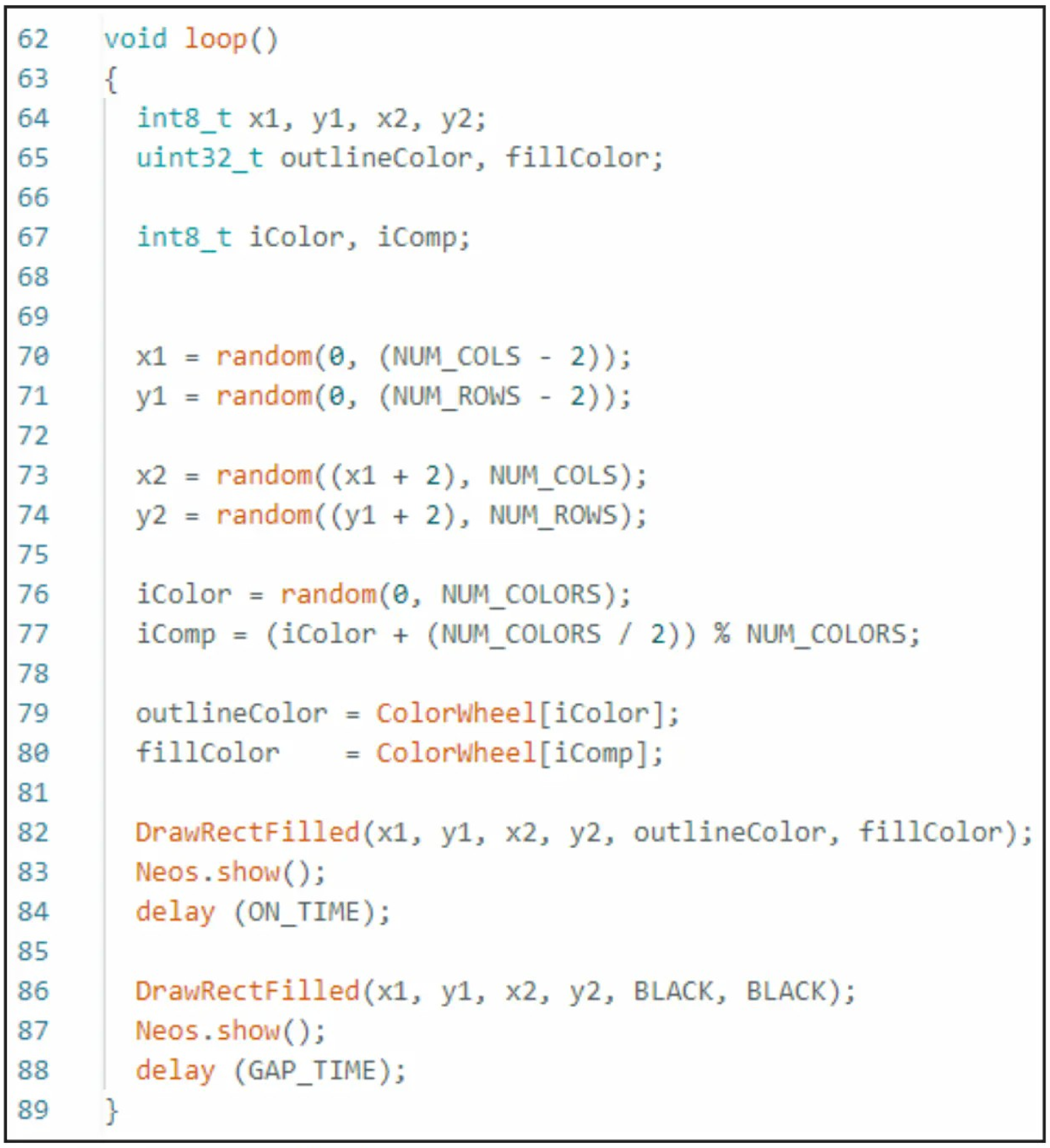
```
// Draw a solid (filled) rectangle between (x1, y1) and (x2, y2)
void DrawRectSolid (int8_t x1, int8_t y1, int8_t x2, int8_t y2, uint32_t color)
{
  for (int8_t y = y1; y <= y2; y++)
  {
    DrawLine(x1, y, x2, y, color);
  }
}
```

Listing 6(a): our DrawRectSolid() function.



```
// Draw a rectangle between (x1, y1) and (x2, y2)
// Use one color for the outline and one for the fill
void DrawRectFilled (int8_t x1, int8_t y1, int8_t x2, int8_t y2,
                     uint32_t outlineColor, uint32_t fillColor)
{
  DrawRectOutline(x1, y1, x2, y2, outlineColor);
  DrawRectSolid((x1+1), (y1+1), (x2-1), (y2-1), fillColor);
}
```

Listing 7(a): our DrawRectFilled() function.



```
62  void loop()
63  {
64    int8_t x1, y1, x2, y2;
65    uint32_t outlineColor, fillColor;
66
67    int8_t iColor, iComp;
68
69
70    x1 = random(0, (NUM_COLS - 2));
71    y1 = random(0, (NUM_ROWS - 2));
72
73    x2 = random((x1 + 2), NUM_COLS);
74    y2 = random((y1 + 2), NUM_ROWS);
75
76    iColor = random(0, NUM_COLORS);
77    iComp = (iColor + (NUM_COLORS / 2)) % NUM_COLORS;
78
79    outlineColor = ColorWheel[iColor];
80    fillColor    = ColorWheel[iComp];
81
82    DrawRectFilled(x1, y1, x2, y2, outlineColor, fillColor);
83    Neos.show();
84    delay (ON_TIME);
85
86    DrawRectFilled(x1, y1, x2, y2, BLACK, BLACK);
87    Neos.show();
88    delay (GAP_TIME);
89  }
```

Listing 7(b): our updated **loop()** function for drawing random boxes on the 'screen'.

'bottom of the pile', our **DrawPixel()** function calls our **GetNeoNum()** function (we'll see why this is important shortly).

We are basing each new function on one or more existing functions that have already been tested and verified, thereby making our lives a lot easier.

Our updated **loop()** function is illustrated in Listing 7(b). Observe lines 82 and 86, where we've swapped out our previous calls to the **DrawRectSolid()** function with calls to our new **DrawRectFilled()** function and added the second colour argument to each.

I decided to use complementary colour pairs for the outline and fill colours. On line 76, we generate a random index value between 0 and 11, then on line 77, we generate the index value for the complementary colour to this using the equation we described earlier. All I can say is that the result looks "awesome" (and you can quote me on that)!

Resource limitations

This rest of this column is going to be a bit heavy, so we will take things step-by-step and simplify things a little, in that we will consider only mainstream microcontroller and compiler implementations. Otherwise, we will find ourselves going down a rabbit hole.

A microcontroller unit (MCU) is driven by a system clock signal. This may range from a few kilohertz (thousands of hertz) to hundreds of megahertz (hundred millions of hertz), where 1Hz (hertz) is one cycle per second.

Generally, we want our programs to run as fast as they can. Sometimes, however, we need to make trade-offs, because doing something fast might consume more memory.

One way to visualise this is as a triangle with the vertices representing clock cycles, data memory (SRAM) and program memory (flash), as in Fig.6. All these are finite resources—we can't exceed the maximum of any of them.

All MCUs contain an arithmetic logic unit (ALU) that can perform simple operations on integers. These include logical operations like AND, OR and XOR, mathematical operations like ADD and SUB (subtract), and other operations like CMP (compare).

CMP may be thought of as a hybrid instruction that combines logical and mathematical attributes. It subtracts two values, which is a mathematical operation, but it doesn't store the result. Instead, it updates the processor's status flags, like the zero, carry, negative and overflow flags.

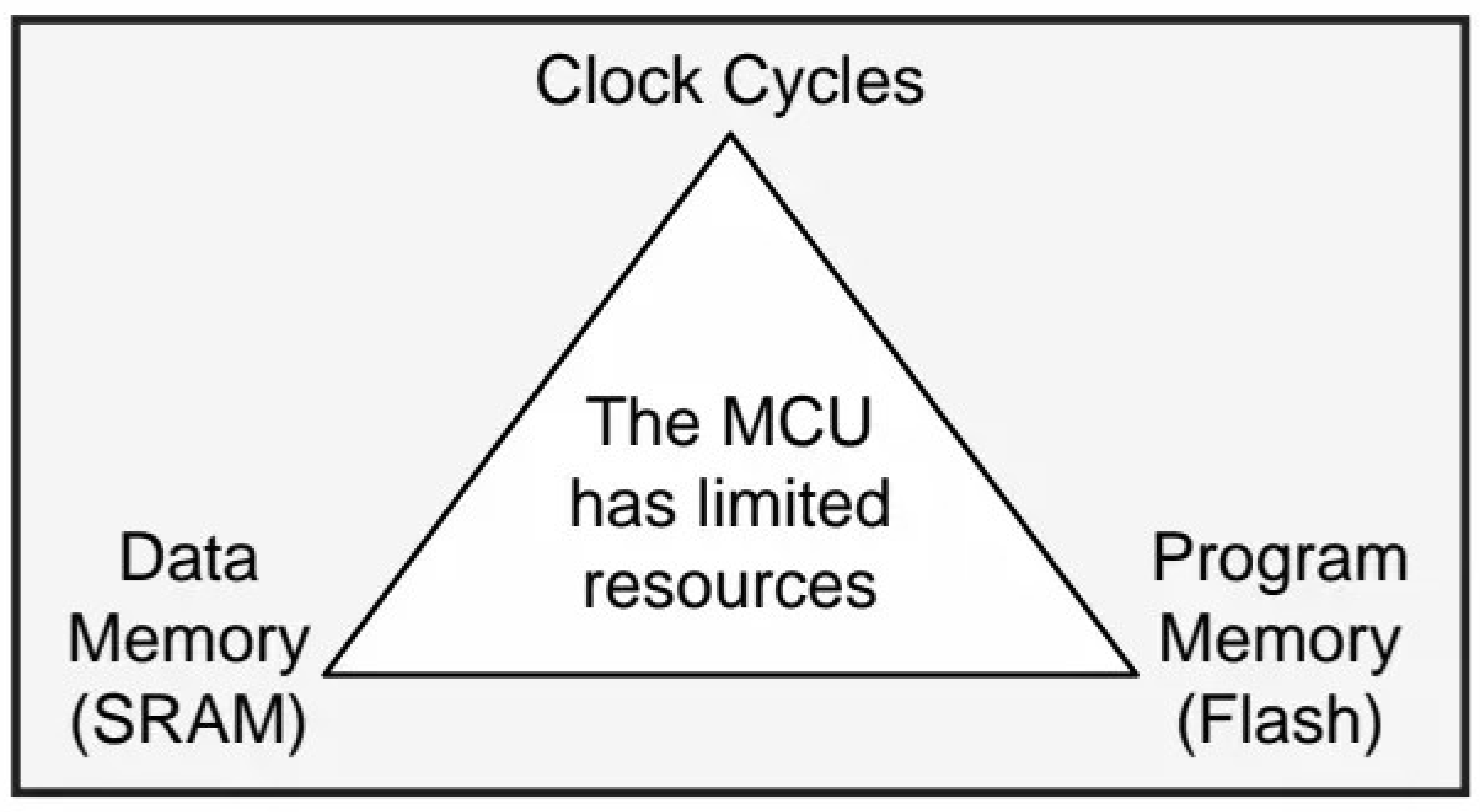


Fig.6: trade-offs of limited MCU resources.

The processor subsequently employs logical operations in conjunction with these flags to make determinations (is A equal to B) and decisions (if A is equal to B, then…).

Some microcontrollers have 8-bit (one byte) data paths and ALUs that operate on such values. Others have 16-bit (two byte) data paths and ALUs. Some have 32-bit (four byte) data paths and ALUs, while a few go beyond that.

If your integers are **x** bits wide and you are running on an **x**-bit machine, performing an operation like AND or ADD will consume only a single clock cycle. However, if your integers are 16 bits wide and you are running on an 8-bit machine, then performing an operation like AND or ADD will usually consume two clock cycles (one for each byte).

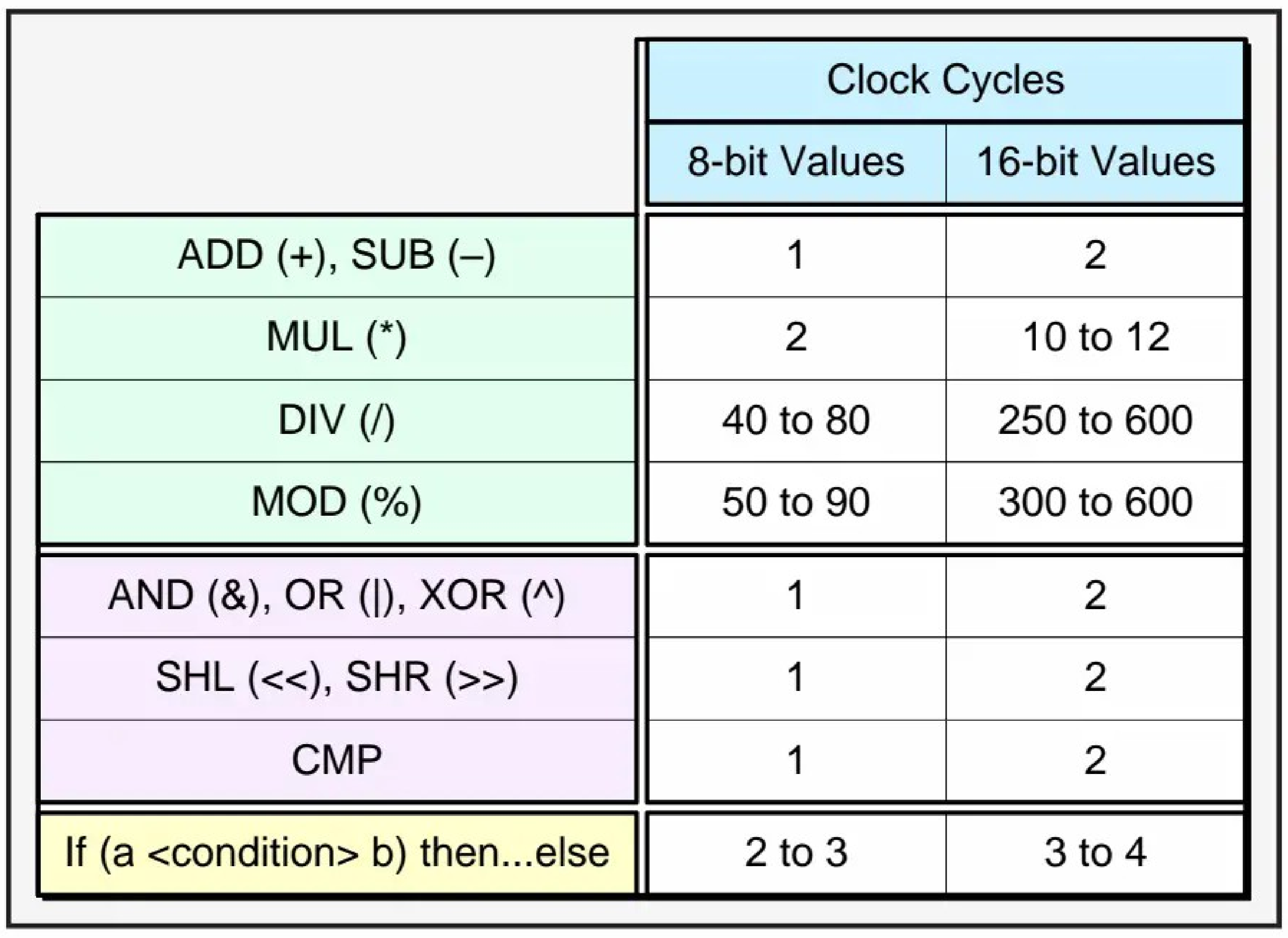
As previously discussed, the Arduino Uno R3 has an **int** of 16 bits (two bytes) and a **long** of 32 bits (four bytes), but the CPU is an 8-bit model, so operations like AND or ADD on these types will take two and four clock cycles, respectively.

Instructions like MUL (multiply) and DIV (divide) can be implemented using lots of simpler operations, like shift, add and subtract. To speed things up, some MCUs contain a hardware multiplier and some contain both a multiplier and a divider. Others have to resort to software routines to do all the work (which is slower).

The ATmega328P microcontroller used in the Arduino Uno includes an 8-bit hardware multiplier. This allows it to perform a single-cycle 8-bit by 8-bit multiplication and store the 16-bit result in two 8-bit registers in two clock cycles.

Multiplying two 16-bit integers on the ATmega328P requires breaking the operation into multiple 8-bit by 8-bit multiplications. The complete 16-bit multiplication typically takes around 10–12 clock cycles.

The ATmega328P does not have a hardware divider, meaning division must be implemented in software using repeated subtraction, bit-shifting, or some other algorithm. As a result, an 8-bit ÷ 8-bit division can consume 40–80 clock cycles, while a 16-bit ÷ 16-bit division can suck up 250–600 clock cycles (depending on the operand values etc).



	Clock Cycles	
	8-bit Values	16-bit Values
ADD (+), SUB (–)	1	2
MUL (*)	2	10 to 12
DIV (/)	40 to 80	250 to 600
MOD (%)	50 to 90	300 to 600
AND (&), OR (|), XOR (^)	1	2
SHL (<<), SHR (>>)	1	2
CMP	1	2
If (a <condition> b) then...else	2 to 3	3 to 4

Fig.7: instructions vs. clock cycles on the ATmega328.

Worst of all are modulo operations. Since performing a modulo requires a division operation plus a multiplication and subtraction, it takes a bit longer than division.

Rooting around the ATmega328P's data sheet (https://pemag.au/link/ac4x), coupled with a little investigative reporting, reveals some interesting numerical factoids that I've summarised in Fig 7.

An **if()** statement requires a CMP operation followed by a conditional branch. The conditions can include **==**, **!=**, **<**, **>**, **<=** and **>=**. The conditional branch will consume one clock cycle if the branch is not taken or two if it is.

Testing times

As we noted, the ATmega328P microcontroller powering the Arduino Uno R3 is an 8-bit device. It has 2kiB of RAM (data memory) and 32kiB of flash (program memory). The system clock runs at 16MHz by default. These numbers establish the boundaries of our resource triangle in Fig.6.

The utility function at the conceptual 'bottom of the pile'—the one that's ultimately called by all our other graphics utility functions—is our old friend, **GetNeoNum()**. As you may recall, we originally implemented this using 16-bit integer data types. We also used the modulo operator. Based on the table shown in Fig.7, I think we are justified in saying, "Eeek!".

I just created a simple test program (in the file named **CB-May25-Code-08.txt**) that has everything stripped out apart from key definitions, instantiations, global variables and our **setup()**, **loop()** and **GetNeoNum()** functions. As a reminder, the **GetNeoNum()** function is shown in Listing 8(a). The rationale behind the overall architecture of this function was discussed in the March 2025 issue.

When we first created this function, the way it performed seemed quite spiffy. With our newfound knowledge, we might start looking at it with a more jaundiced eye.

Although we might turn our noses up at a 16MHz clock, that's still sixteen million clock cycles in every second. That's a lot of clock cycles and, to be honest, I haven't noticed any lag in the response of my array.

On the other hand, anything we can do that manages to both minimise memory and increase performance must be a good thing. If we do this now, we will reap the benefits in the future when we create our games.

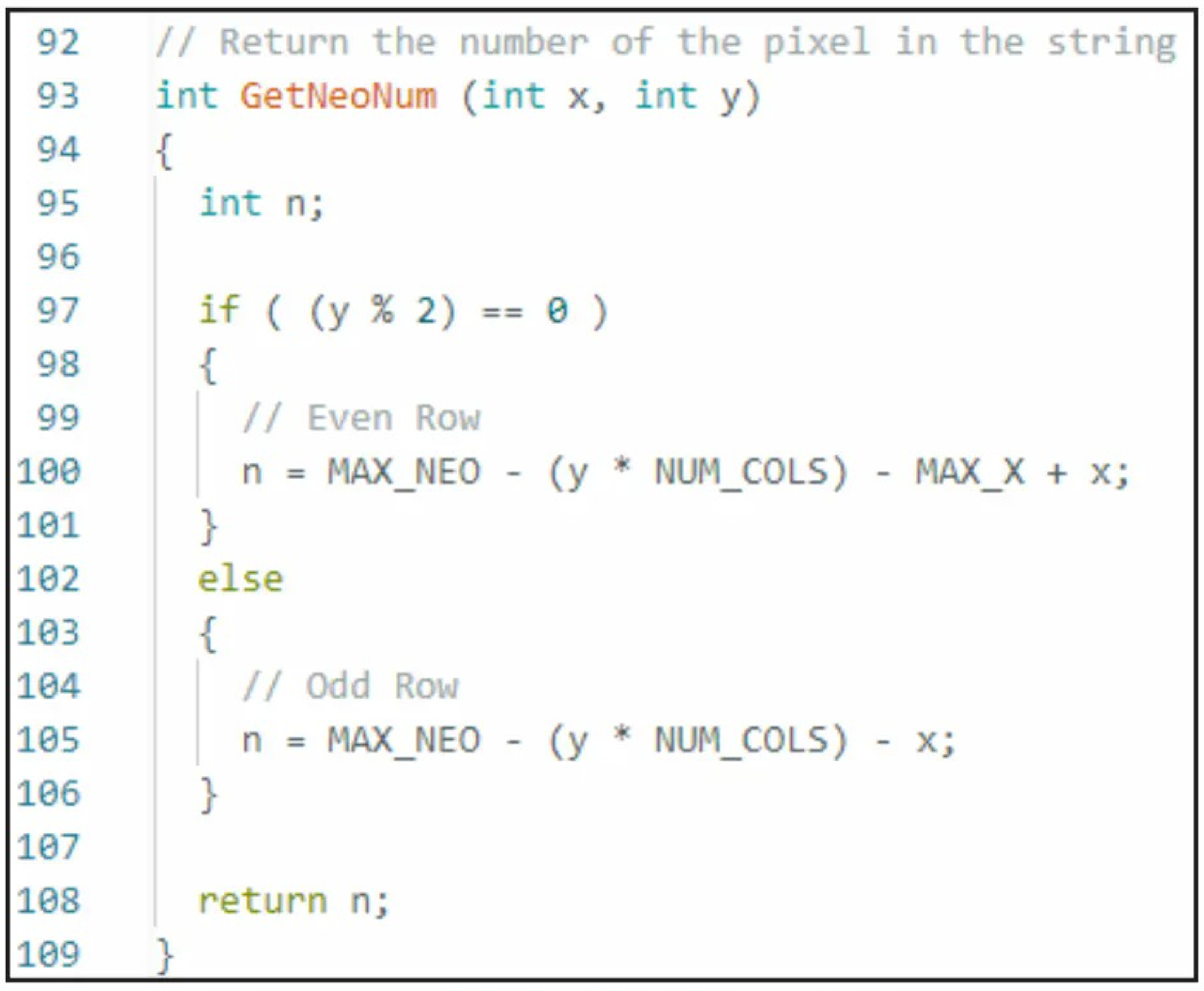
Except for the addition of a global integer variable called **CycleNum** that is initialised to 0, there's nothing in the definitions and suchlike that we haven't seen before. The test itself is found in the **loop()** function as seen in Listing 8(b).

The bulk of the action takes place in the nested loops between lines 67 and 75. This is where we cycle through every pixel in the array, setting the pixel colours.

What's with line 72, where we generate the colour? In the past, I've had tussles when trying to benchmark my code with the Arduino's compiler. Besides regular cunning optimisations, if it thinks that any of the things in your code aren't doing anything useful, there's a chance it will eliminate them.

So, I'm using line 72 to generate a varying index that will access a subset of colours (0 to 7) in our colour wheel, where the colours will change every time we cycle through the loop.

As an aside, if you are interested in learning more about compilers, there's an awesome book called

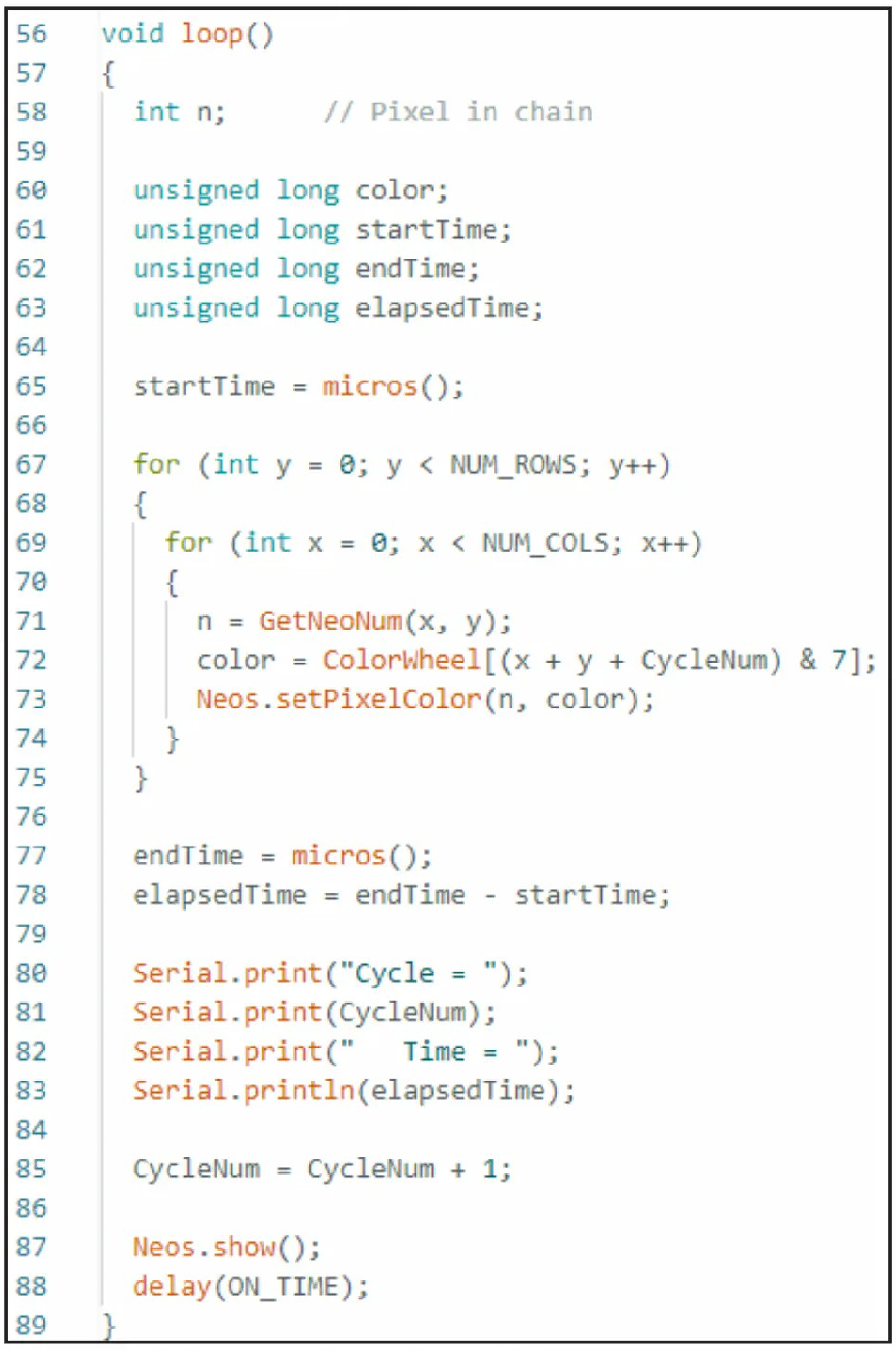


```
// Return the number of the pixel in the string
int GetNeoNum (int x, int y)
{
  int n;

  if ( (y % 2) == 0 )
  {
    // Even Row
    n = MAX_NEO - (y * NUM_COLS) - MAX_X + x;
  }
  else
  {
    // Odd Row
    n = MAX_NEO - (y * NUM_COLS) - x;
  }

  return n;
}
```

Listing 8(a): our original **GetNeoNum()** using **int**s.



```
56  void loop()
57  {
58    int n;      // Pixel in chain
59
60    unsigned long color;
61    unsigned long startTime;
62    unsigned long endTime;
63    unsigned long elapsedTime;
64
65    startTime = micros();
66
67    for (int y = 0; y < NUM_ROWS; y++)
68    {
69      for (int x = 0; x < NUM_COLS; x++)
70      {
71        n = GetNeoNum(x, y);
72        color = ColorWheel[(x + y + CycleNum) & 7];
73        Neos.setPixelColor(n, color);
74      }
75    }
76
77    endTime = micros();
78    elapsedTime = endTime - startTime;
79
80    Serial.print("Cycle = ");
81    Serial.print(CycleNum);
82    Serial.print("   Time = ");
83    Serial.println(elapsedTime);
84
85    CycleNum = CycleNum + 1;
86
87    Neos.show();
88    delay(ON_TIME);
89  }
```

Listing 8(b): our test sequence using **int**s.

Writing a C Compiler by Nora Sandler and published by No Starch Press (https://nostarch.com/writing-c-compiler). However, be advised that this is a meaty morsel that will make your brain ache (well, it did mine).

The Arduino's `micros()` function returns the number of microseconds (millionths of a second) since the program started running. We call this function on lines 65 and 77, and use the returned values in line 78 to determine the time taken to cycle through all the pixels.

Oodles of optimisations

We'll start by generating an 'as-is' baseline (it can only get faster from here). When we look at the instruction-clock cycle combos in Fig.7, coupled with our `GetNeoNum()` function in Listing 8(a), we see that there's an elephant in the room and a fly in the ointment (I never metaphor I didn't like). That is our use of the modulo operator on line 97, where we wrote, `if ((y % 2) == 0)`.

All we are using this operator for is to determine if we are dealing with an odd or even row. Essentially, we are dividing the `y` value (the row number) by 2 and looking at the remainder (the bit that 'drops off the end').

However, all we really need to do is check to see if the least-significant bit of the `y` value is 0 or 1. We can achieve the same effect by saying `if ((y & 1) == 0)`. Since we are currently working with 16-bit integers, our `&` operation will consume 2 clock cycles, as compared to the `%` operation which consumed a whopping 300–600 clock cycles.

I just made this change in the file named **CB-May25-Code-09.txt**.

Next, we can change all of our `int` variables (16 bits) into fixed-width `int8_t` data types (8 bits). This includes the return type of the `GetNeoNum()` function itself. I did that in the file named **CB-May25-Code-10.txt**.

As seen in Fig.8, the results are… interesting. Remember that 1µs is a millionth of a second and our system clock is running at 16MHz, so that means we have 16 clock cycles in each 1µs.

Our baseline is… well, what it is. We can't really draw any conclusions at this stage. The reason there's a range of time values is that the code performs different combinations of operations on each pass through the main loop.

Now look at the results from our first optimisation attempt. What happens when we replace the `((y % 2) == 0)` with `((y & 1) == 0)` in our `GetNeoNum()` function? The answer is… absolutely nothing! How can this be after the big buildup I've given you?

The answer is in the Arduino's clever compiler. When it saw the `((y % 2) == 0)` in our original code, it said "Ah ha!" to itself. Since the compiler could see that this operation never changed, it replaced it with `((y & 1) == 0)` without our knowing. As a result, our first two tests are identical (which explains the identical results).

Our next test, where we replace 16-bit `int` variables with their `int8_t` counterparts, does provide some improvements in both program size and runtime (phew!).

I really wanted to see what the effect the `%` operator would have. So, I added one more test (in the file named **CB-May25-Code-11.txt**). On line 72 in the main `loop()` function, I replaced our original `(x + y + CycleNum) & 7` with `(x + y + CycleNum) % 8`, which will give the same functional result.

Since the values associated with this operation change each time through the loop, the compiler has no choice but to implement the `%` operator as written. Wow! The code has increased by 68 bytes (not too bad), but the processing time has increased by a whopping 1612 to 1640 clock cycles! And this is with our new 8-bit integers.

Now I want to know how bad it would be with our original 16-bit integers (I will run that test and share the results in a future column). Also, I have some ideas for more optimisations. Why don't you noodle on this yourself and we'll compare notes later?

Tricky triadics

Earlier, I tasked you with working out the equations we could use to generate triadic colours as illustrated in Fig.2(c). When you think about it, this is not dissimilar to the way we generated our split complementary colours.

We can determine the index values of our two triadic colours using the expressions `(i + (NUM_COLORS / 3)) % NUM_COLORS` and `(i + ((NUM_COLORS * 2) / 3)) % NUM_COLORS`.

From our previous discussions, we already know that, starting with our original colour (cyan in this example), we always want to generate the index values for our new colours by travelling clockwise around our colour wheel and then using the `%` (modulo) operator to restrict the results to our 0 to 11 allowable index values.

Based on this, the first of these equations is relatively obvious. We start by adding `(NUM_COLORS / 3)` to our current index value, which will take us ⅓ (120°) clockwise around the colour wheel.

To generate the second value, we need to travel ⅔ (240°) around the colour wheel in the clockwise direction. Our knee jerk reaction might be to calculate this as `((NUM_COLORS / 3) * 2)`; that is, divide by three to generate the 120° value, then multiply by two to generate the desired 240° value.

Instead, I decided to use `((NUM_COLORS * 2) / 3)`, multiplying by two to generate a 720° value and then dividing by three to generate the desired 240° value. Why? A divide followed by a multiply returns the same result as a multiply followed by a divide… or does it?

The problem is that integer divisions in C/C++ discard (truncate) any remainder. In our current implementation, we have 12 colours, so either scenario will work because 12 can be divided by 3 without any remainder.

For the sake of argument, though, suppose we decided to use 14 colours in the future. In this case, 14 ÷ 3 = 4.66, which would be truncated to 4 and 4 × 2 = 8. By comparison, 14 × 2 = 28 and 28 ÷ 3 = 9.33, which will be truncated to 9.

Nine is closer to where we want to be than eight. The point is that, when working with integers, we typically try to perform multiplications (that make things bigger) before divisions (that make things smaller) so we don't lose too much information during intermediate steps.

Blushing in shame!

It's happened again. I said that we were going to perform some more experiments with our 8-segment bar graph display, but we simply don't have the space or time. We will do this next month, I promise!

Over to you. If you have any thoughts that you'd care to share on anything you've read here, please feel free to contact me at max@clivemaxfield.com. Until next time, make sure you have a good one!

P=Program D=Data B=Bytes	P-Storage (B)	D-Storage (B)	Time (µs)
Baseline	4178	308	1040 to 1044
Replace % with &	4178	308	1040 to 1044
Replace int with int8_t	4140	308	968 to 972
Add a % in loop() function	4208	308	2580 to 2612

Fig.8: the results of my attempts at speed optimisation of the code.

Useful Bits and Pieces from Our Arduino Bootcamp series

Arduino Uno R3 microcontroller module	https://pemag.au/link/ac2g
Solderless breadboard	https://amzn.to/3O2L3e8
8-inch (20cm) jumper wires (male-to-male)	https://amzn.to/3O4hnxk
Long-tailed 0.1-inch (2.54mm) pitch header pins	https://pemag.au/link/ac2h
LEDs (assorted colours)	https://amzn.to/3E7VAQE
Resistors (assorted values)	https://amzn.to/3O4RvBt
Ceramic capacitors (assorted values)	https://pemag.au/link/ac2i
16V 100µF electrolytic capacitors	https://pemag.au/link/ac2j
Momentary pushbutton switches	https://amzn.to/3Tk7Q87
Kit of popular SN74LS00 chips	https://pemag.au/link/ac2k
74HC595 8-bit shift registers	https://pemag.au/link/ac1n

Other stuff

Soldering guide	https://pemag.au/link/ac2d
Basic multimeter	https://pemag.au/link/ac2f

Components for Weird & Wonderful Projects, part 1

4-inch (10cm) jumper wires (optional)	https://pemag.au/link/ac2l
8-segment DIP red LED bar graph displays	https://pemag.au/link/ac2c

Components for Weird & Wonderful Projects, part 2

144 tricolour LED strip **(required)**	https://pemag.au/link/ac2s
Pushbuttons (assorted colours) **(required)**	https://pemag.au/link/ac2t
7-Segment display modules **(recommended)**	https://pemag.au/link/ac2u
2.1mm panel-mount barrel socket **(optional)**	https://pemag.au/link/ac2v
2.1mm inline barrel plug **(optional)**	https://pemag.au/link/ac2w
25-way D-sub panel-mount socket **(optional)**	https://pemag.au/link/ac2x
25-way D-sub PCB-mount plug* **(optional)**	https://pemag.au/link/ac2y
15-way D-sub panel-mount socket **(optional)**	https://pemag.au/link/ac2z
15-way D-sub PCB-mount plug^ **(optional)**	https://pemag.au/link/ac30

* one required for each game cartridge you build
^ one required for each auxiliary control panel you build

Components for Weird & Wonderful Projects, part 3

22 AWG multicore wire kit	https://pemag.au/link/ac3m
20 AWG multicore wire kit	https://pemag.au/link/ac3n

Components for Weird & Wonderful Projects, part 4

5V Buck converter module	https://pemag.au/link/ac2m
9V 2A power supply **(optional)**	https://pemag.au/link/ac46
Bench power supply **(optional)**	https://pemag.au/link/ac48

I have no idea why everyone calls me "Mad Max"!

PoLabs
For more information or software download please visit
www.poscope.com/epe

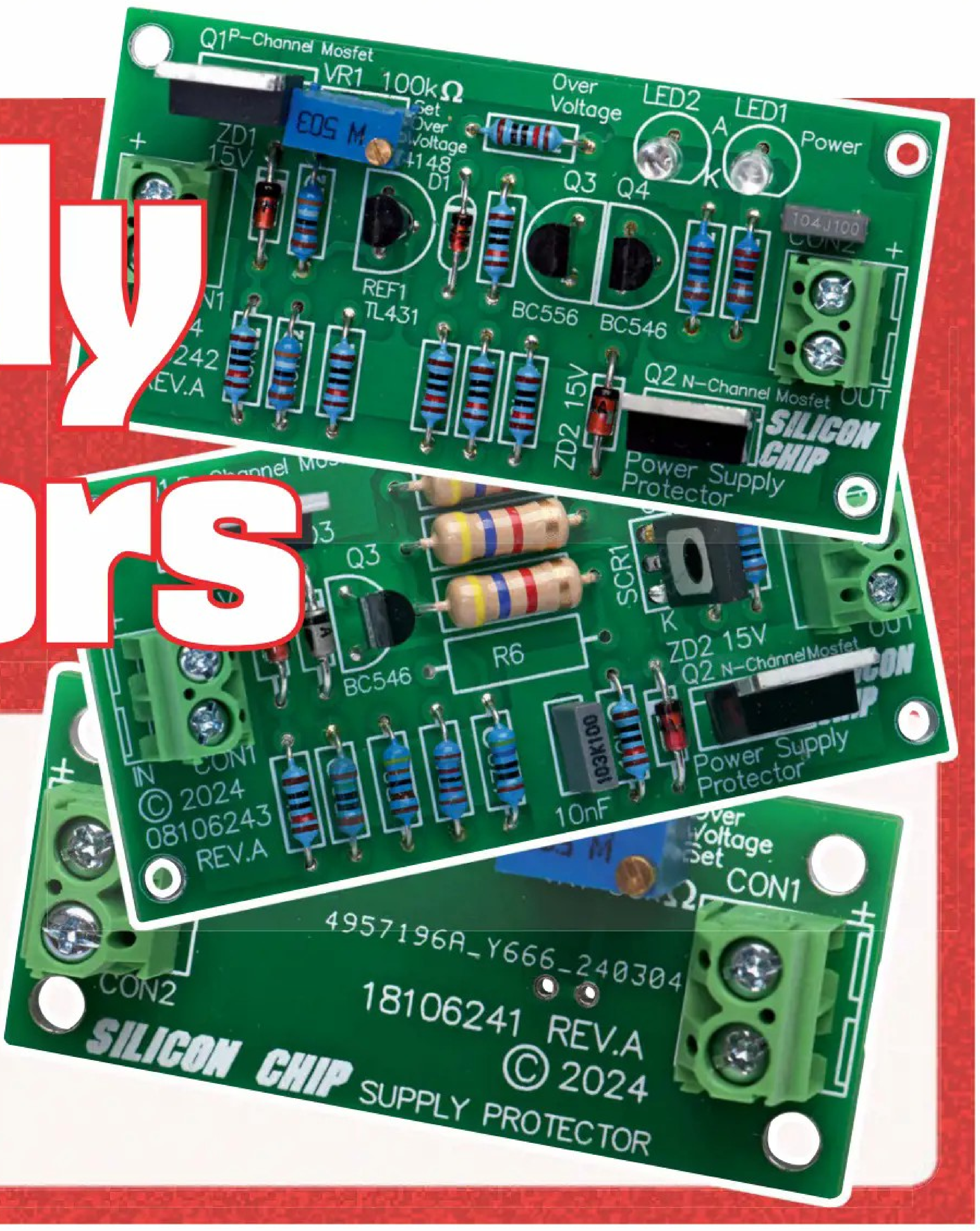
PoKeys
Connect, Control
- USB
- Ethernet
- Web server
- Modbus
- CNC (Mach3/4)
- IO
- PWM
- Encoders
- LCD
- Analog inputs
- Compact PLC

Stepper motor drivers
Drive
- up to 256 microsteps
- 50 V / 6 A
- USB configuration
- Isolated
- up to 32 microsteps
- 30 V / 2.5 A

PoScope Mega1+
PoScope Mega50
Measure
- up to 50MS/s
- resolution up to 12bit
- Lowest power consumption
- Smallest and lightest
- 7 in 1: Oscilloscope, FFT, X/Y, Recorder, Logic Analyzer, Protocol decoder, Signal generator

Project by John Clarke

DC Supply Protectors


Q1 P-Channel Mosfet
VR1 100kΩ
Over Voltage
LED2
LED1
Power
ZD1 15V
REF1 TL431
BC556
BC546
Q3
Q4
Q2 N-Channel Mosfet
SCR1
R6
ZD2 15V
CON1
CON2
OUT
IN
SILICON CHIP
Power Supply Protector
© 2024
08106243 REV.A
18106241 REV.A
SILICON CHIP SUPPLY PROTECTOR

Any one of these three simple, inexpensive circuits will protect your equipment from damage due to an incorrectly connected or malfunctioning power supply. They protect against a higher than expected voltage or a reverse polarity supply and have very little effect on the voltage applied to the device.

Many devices are powered using a mains plugpack or power 'brick'. All is well if you use the proper supply and it is wired correctly. However, damage can occur if the wrong supply is used or it is miswired, applying either the wrong polarity voltage or an excessively high voltage to the item to be powered.

That is an especially big problem if you haven't used the device for many years, have moved, if you've had to buy a replacement power supply (due to failure or loss), or someone else is using it who is not familiar with the correct supply.

Our Supply Voltage Protectors prevent damage to equipment in the case of an incorrect input voltage. They switch off power to the equipment if the input voltage is too high and prevent current flow if the polarity is incorrect.

A supply that produces more voltage than a piece of equipment expects can damage its internal components. Applying reverse polarity to a circuit can also irreparably damage parts, such as ICs and electrolytic capacitors unless the circuitry already includes reverse polarity protection.

Such protection (eg, a diode) often reduces the voltage available to the device. However, the designs presented here are different, as they use a Mosfet instead that loses very little (basically no) voltage.

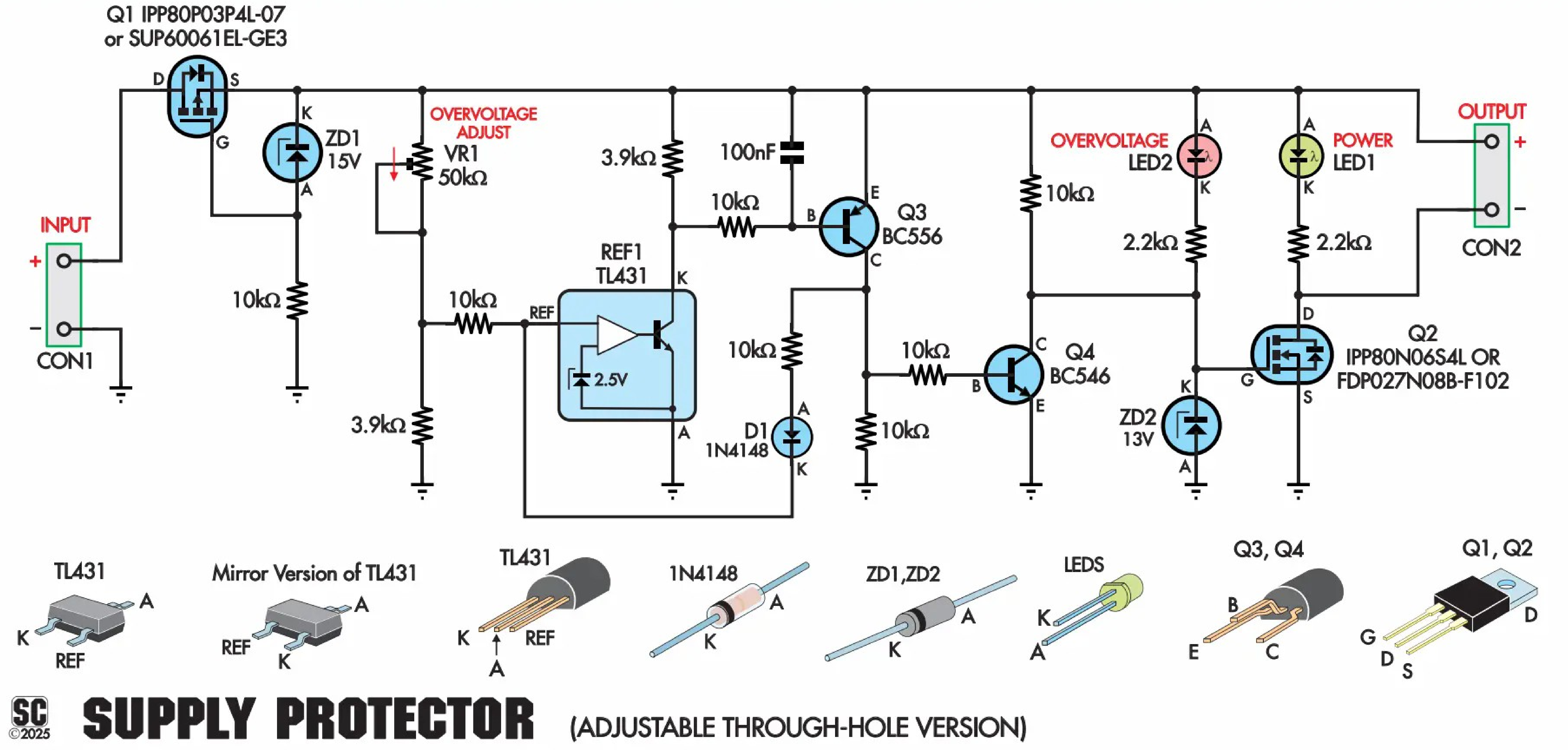

Q1 IPP80P03P4L-07
or SUP60061EL-GE3
D
S
G
K
ZD1 15V
A
INPUT
+
–
CON1
10kΩ
OVERVOLTAGE ADJUST
VR1 50kΩ
10kΩ
3.9kΩ
REF
REF1 TL431
K
A
2.5V
3.9kΩ
100nF
10kΩ
B
E
C
Q3 BC556
10kΩ
D1 1N4148
A
K
10kΩ
10kΩ
B
C
E
Q4 BC546
10kΩ
OVERVOLTAGE
LED2
A
K
2.2kΩ
POWER
LED1
A
K
2.2kΩ
ZD2 13V
K
A
D
G
S
Q2 IPP80N06S4L OR FDP027N08B-F102
OUTPUT
+
–
CON2
TL431
K
REF
A
Mirror Version of TL431
REF
K
A
TL431
K
REF
A
1N4148
K
A
ZD1,ZD2
K
A
LEDS
K
A
Q3, Q4
B
E
C
Q1, Q2
G
D
S
D
SC ©2025
SUPPLY PROTECTOR
(ADJUSTABLE THROUGH-HOLE VERSION)

Fig.1: the adjustable TH version uses Mosfet Q1 for reverse-polarity protection and Q2/REF1 for over-voltage protection.

DC Supply Protectors

These Protectors can be built as standalone devices. Still, as they are relatively compact and inexpensive, they should ideally be installed within existing equipment. That way, nothing bad should happen unless you connect a supply that's outside the usual ratings.

We are presenting three versions of the circuit. Two have a trimpot adjustment to set the overvoltage protection threshold; both can be used with plugpacks that produce up to 27V DC.

One of those two versions uses surface-mounting parts, so it is smaller than the other versions and can be squeezed into tighter spaces. It is rated to handle up to 3A. This version can have the overvoltage set as low as 3V. The through-hole equivalent can handle more current, up to 7A. However, it needs at least 5V to operate.

The third version is slightly cheaper to build but can only be adjusted in voltage steps determined by zener diode reverse breakdown or avalanche values. It can handle up to 50V. One advantage of this version is that it doesn't require a setup procedure; you simply build it, install it and away you go. Its minimum overvoltage protection setting is 7.5V.

Circuit details

Figs.1-3 show the circuits of all three versions. They all provide reverse-polarity protection using P-channel Mosfet Q1. With a correct polarity connection, Q1 conducts initially via the intrinsic diode within the Mosfet, allowing current to flow and voltage to appear at the source.

The 10kΩ resistor then pulls the gate to ground, so Q1's channel is switched on. This allows current to bypass the intrinsic diode, greatly reducing the voltage across it. Zener diode ZD1 prevents the gate-to-source voltage from exceeding the maximum specification of 10V for the surface-mount device (SMD) version and 15V for the through-hole (TH) version.

On the other hand, if the polarity of the input supply is reversed, Q1's intrinsic diode is reverse-biased and the gate voltage is the same as the source. So the Mosfet remains off, and no current flows to the load.

Adjustable over-voltage protector circuit

Figs.1 & 2 are the adjustable overvoltage protector circuits. The circuit is the same for the TH (Fig.1) and SMD (Fig.2) versions of the board, except that some parts have different codes/packages and some ½W TH resistors are replaced with two parallel ¼W SMD resistors (where a higher power rating is required).

This overvoltage protection circuitry comprises N-channel Mosfet Q2, shunt regulator IC REF1 and bipolar transistors Q3 & Q4. Q2 is usually held on via gate voltage applied through the 10kΩ resistor from the positive supply. In this case, it has a low resistance between its drain and source, connecting the ground terminal of CON1 to the negative side of CON2, so current can flow between the load and supply.

Zener diode ZD2 protects the gate of Q2 from excessive voltage. A 10V zener is used for the SMD version, while a 13V zener diode is used for the TH version, reflecting the ratings of the selected Mosfet types.

Power indicator LED1 is lit by current flow through the 2.2kΩ resistor to the negative side of the supply. For the SMD version, the 2.2kΩ resistor is instead two 4.7kΩ resistors in parallel, in case the supply voltage is at the higher end of the allowable range.

The TL431 adjustable shunt voltage reference IC, REF1, detects an overvoltage condition from the input supply. Fig.4 shows the circuitry within the TL431, which includes a 2.5V reference, an op amp, an output transistor and a protection diode.

When used as a voltage reference, the REF input connects to the cathode, placing the TL431 in a negative feedback configuration, where it regulates its 'cathode' voltage to match its internal reference voltage. If you want a higher cathode voltage, a voltage divider is included between the cathode and anode, with the divided voltage applied to the REF input.

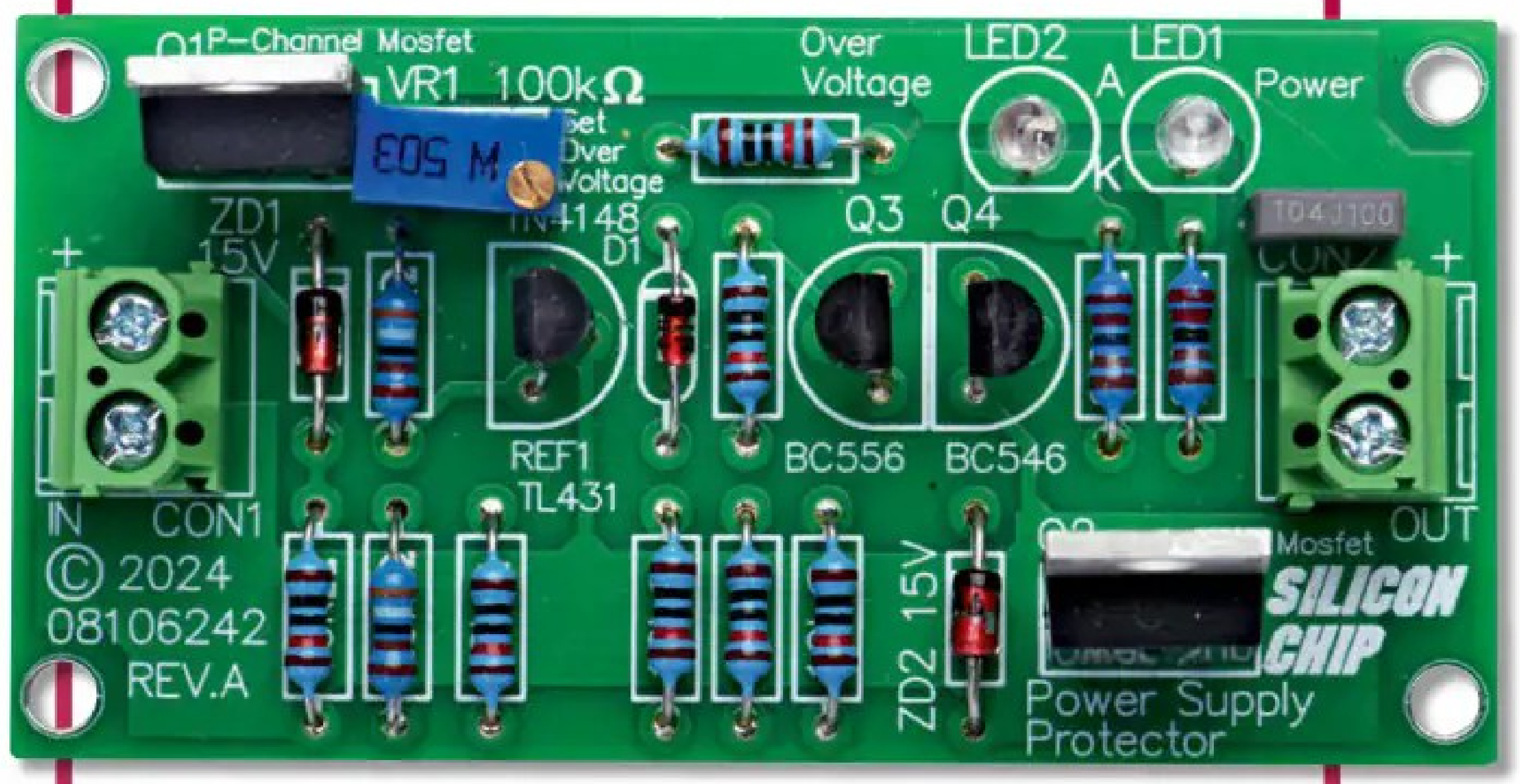
For our circuit, we instead apply a voltage to the REF terminal via a resistance. In this case, the TL431 operates in open-loop mode without any feedback to maintain the reference voltage. This arrangement uses the internal op amp as a comparator, switching the output transistor off if the input voltage (V_{IN}) is lower than the reference voltage, or on otherwise.

When the transistor is off, the cathode connection is pulled to the supply voltage via R_{SUPPLY}. In contrast, when the transistor is switched on, the cathode

Features & Specifications

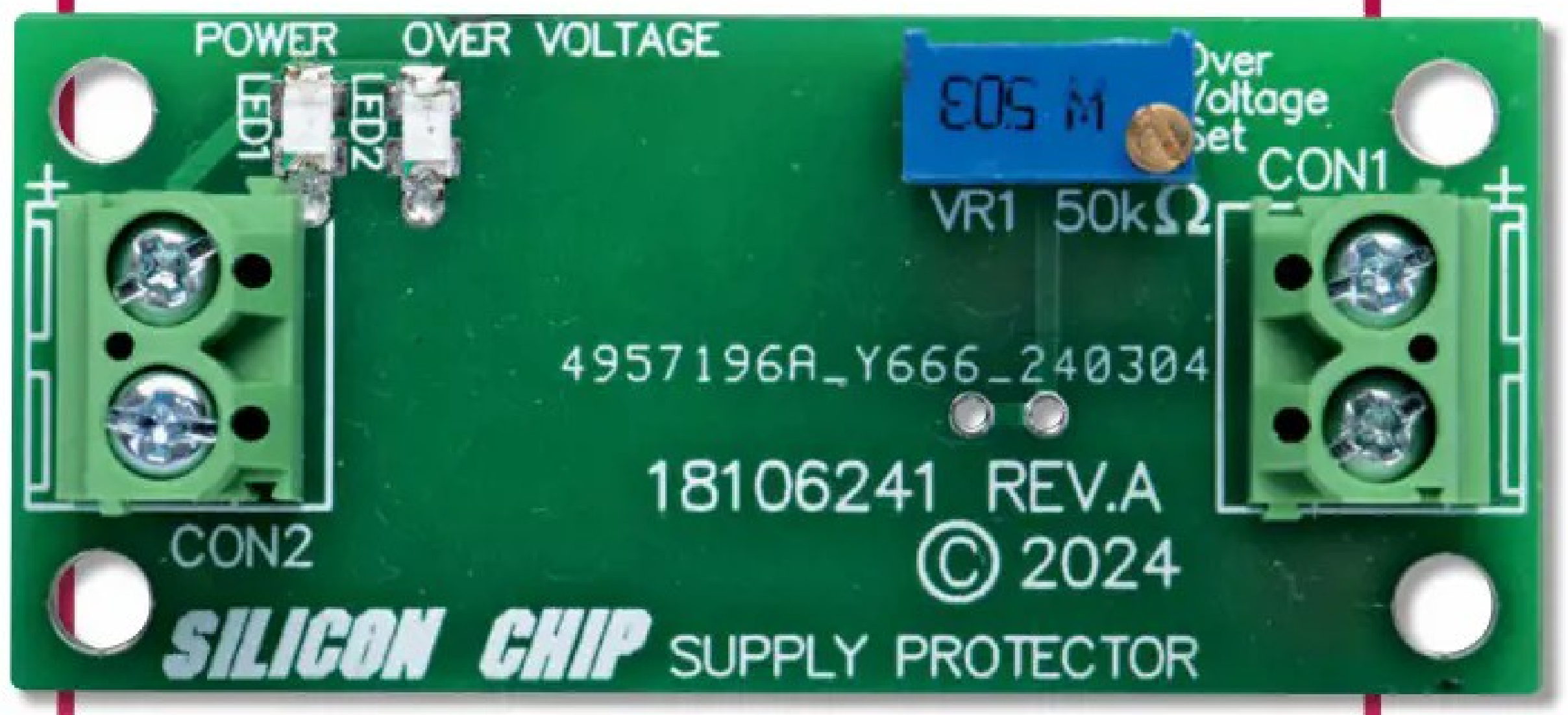
Adjustable through-hole version

- Overvoltage protection threshold: 5-27V *(3-27V if SMD TL431 is used)*
- Input voltage range: 5-27V
- Maximum current: 7A



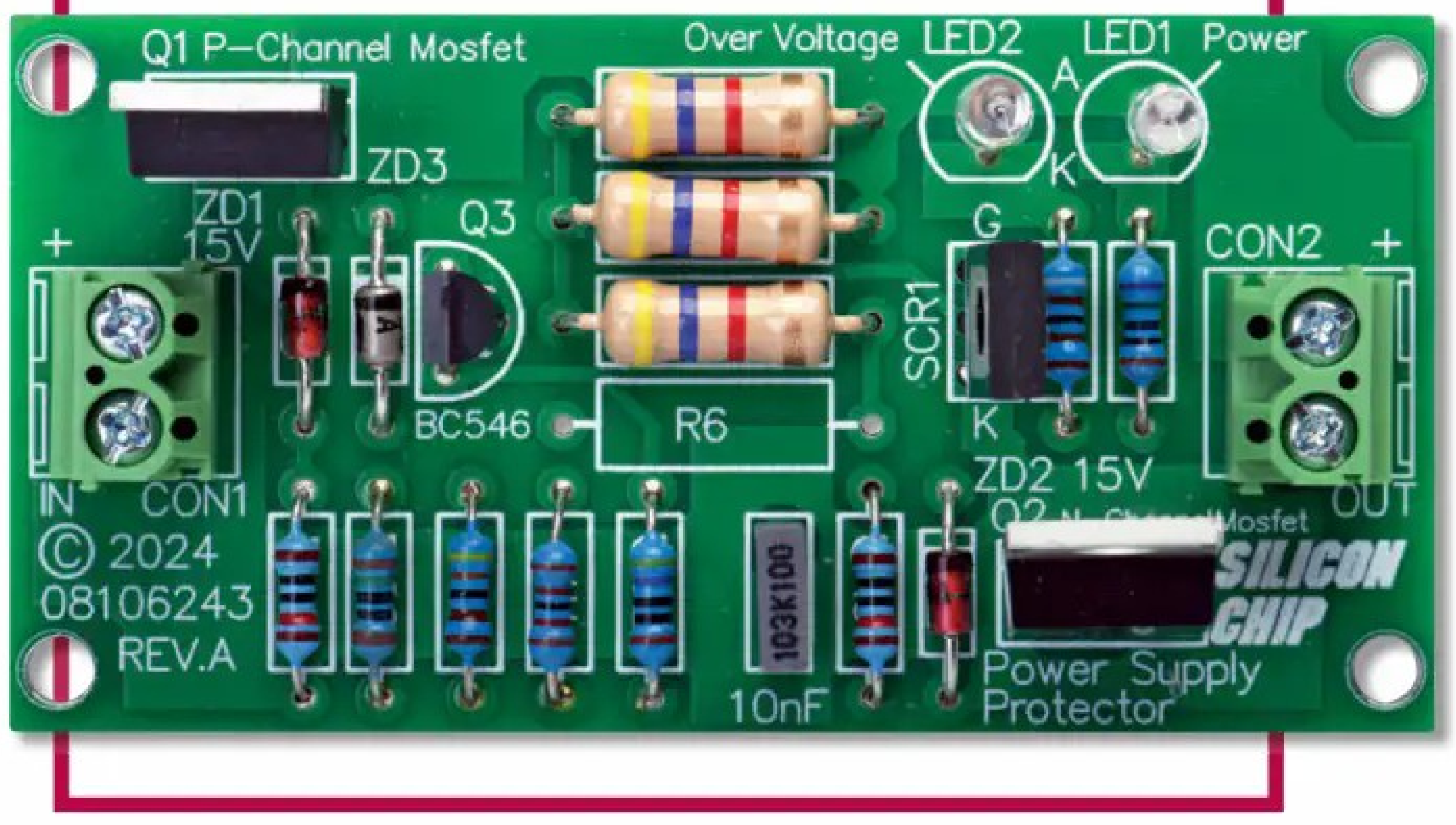
Adjustable SMD version

- Overvoltage protection threshold: 3-27V
- Input voltage range: 3-27V
- Maximum current: 3A



Fixed through-hole version

- Overvoltage protection threshold: 7.5-47.7V
- Input voltage range: 5-50V
- Maximum current: 7A



Kits from Silicon Chip

Adjustable SMD Version (SC6948, £10.50 + P&P): includes the PCB and all onboard parts.

Adjustable TH Version (SC6949, £13.50 + P&P): includes the PCB and all onboard parts with both SOT-23 & TO-92 TL431 ICs.

Fixed TH Version (SC6950, £12 + P&P): comes with the PCB and all onboard parts except ZD3 and R1-R7.

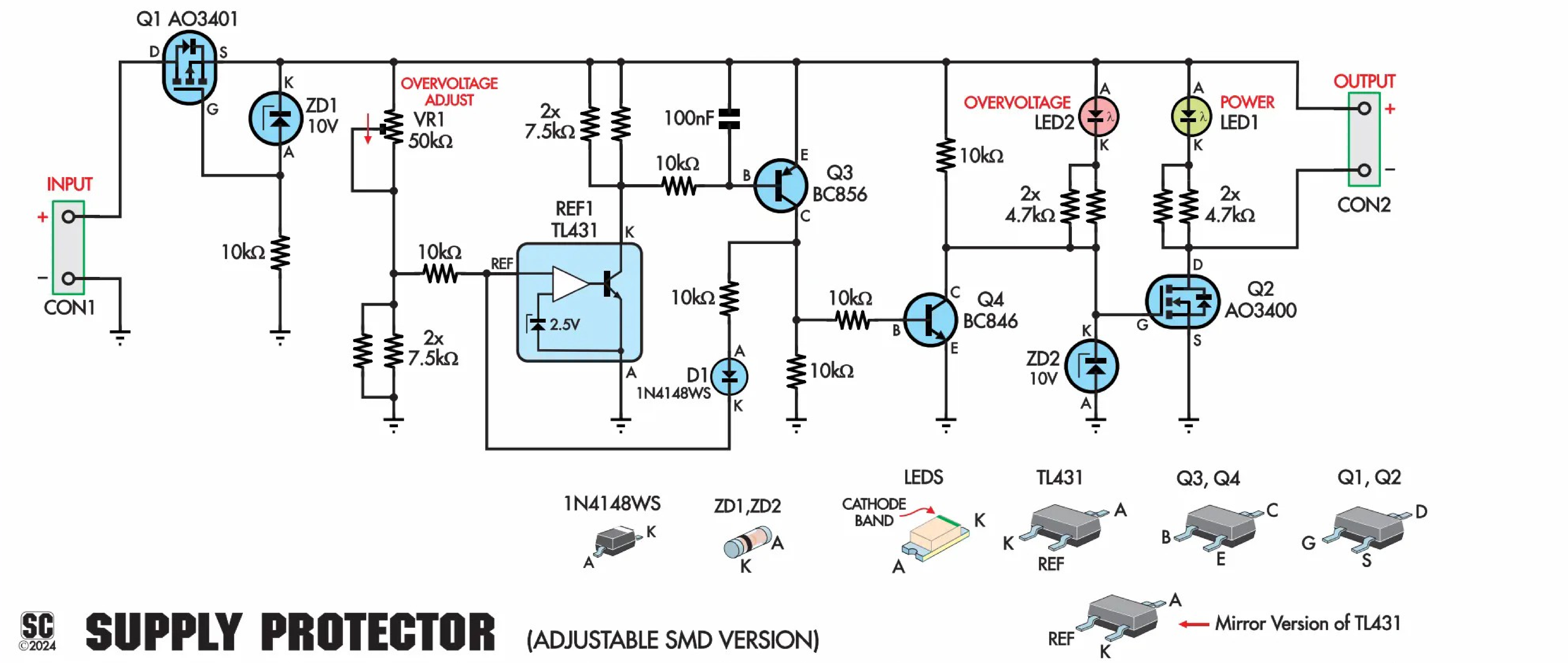

Q1 AO3401
D
S
G
K
ZD1
10V
A
INPUT
+
–
CON1
10kΩ
OVERVOLTAGE ADJUST
VR1
50kΩ
10kΩ
2x 7.5kΩ
2x 7.5kΩ
REF1 TL431
REF
K
A
2.5V
100nF
10kΩ
B
E
C
Q3 BC856
10kΩ
D1
1N4148WS
10kΩ
10kΩ
Q4 BC846
10kΩ
OVERVOLTAGE LED2
2x 4.7kΩ
ZD2 10V
POWER LED1
2x 4.7kΩ
Q2 AO3400
OUTPUT
+
–
CON2
1N4148WS
ZD1,ZD2
LEDS
CATHODE BAND
TL431
Q3, Q4
Q1, Q2
Mirror Version of TL431
SC ©2024
SUPPLY PROTECTOR
(ADJUSTABLE SMD VERSION)

Fig.2: the SMD adjustable version of the circuit is very similar to the one shown in Fig.1. The main differences are the use of alternative devices and the doubling of some resistors for increased power handling.

connection is held close to the anode voltage of 0V.

The state of the output, whether high or low, depends on the voltage applied to the REF input. In our circuit (Fig.1 or Fig.2), this voltage is from the divider connected across the input supply formed by trimpot VR1 and a 3.9kΩ fixed resistor (or two 7.5kΩ resistors in parallel, giving 3.75kΩ).

When this divided (reduced) voltage is below the 2.5V reference, the cathode of REF1 is pulled high. When the divided voltage is above 2.5V, the cathode is pulled low, near to ground potential.

Trimpot VR1, in conjunction with the 3.9kΩ resistor, sets the overvoltage threshold. When the threshold is reached, the cathode of REF1 goes low, so transistor Q3 switches on. It in turn switches on transistor Q4, which pulls the gate of Mosfet Q2 low to switch it off.

In this condition, the high collector voltage of Q3 pulls the adjust terminal of REF1 higher again via diode D1 and the 10kΩ resistor. This provides voltage hysteresis, ensuring that REF1's cathode remains low until the supply voltage drops significantly below the overvoltage setting.

The 100nF capacitor between the base and emitter of Q3 is included to prevent the circuit from initially latching into a voltage overload state at power-up. Immediately after power is applied to CON1, REF1 would momentarily conduct current that would otherwise switch on Q3 and latch REF1 on if it were not for the capacitor momentarily holding Q3 off.

LED2 is the overvoltage indicator and it lights under two conditions. If the input supply exceeds ZD2's breakdown voltage, current will flow through LED2, its 2.2kΩ series resistor and ZD2. However, if the overvoltage threshold is set below ZD2's breakdown voltage, LED2 will only light when there is an overvoltage shutdown, via NPN transistor Q4.

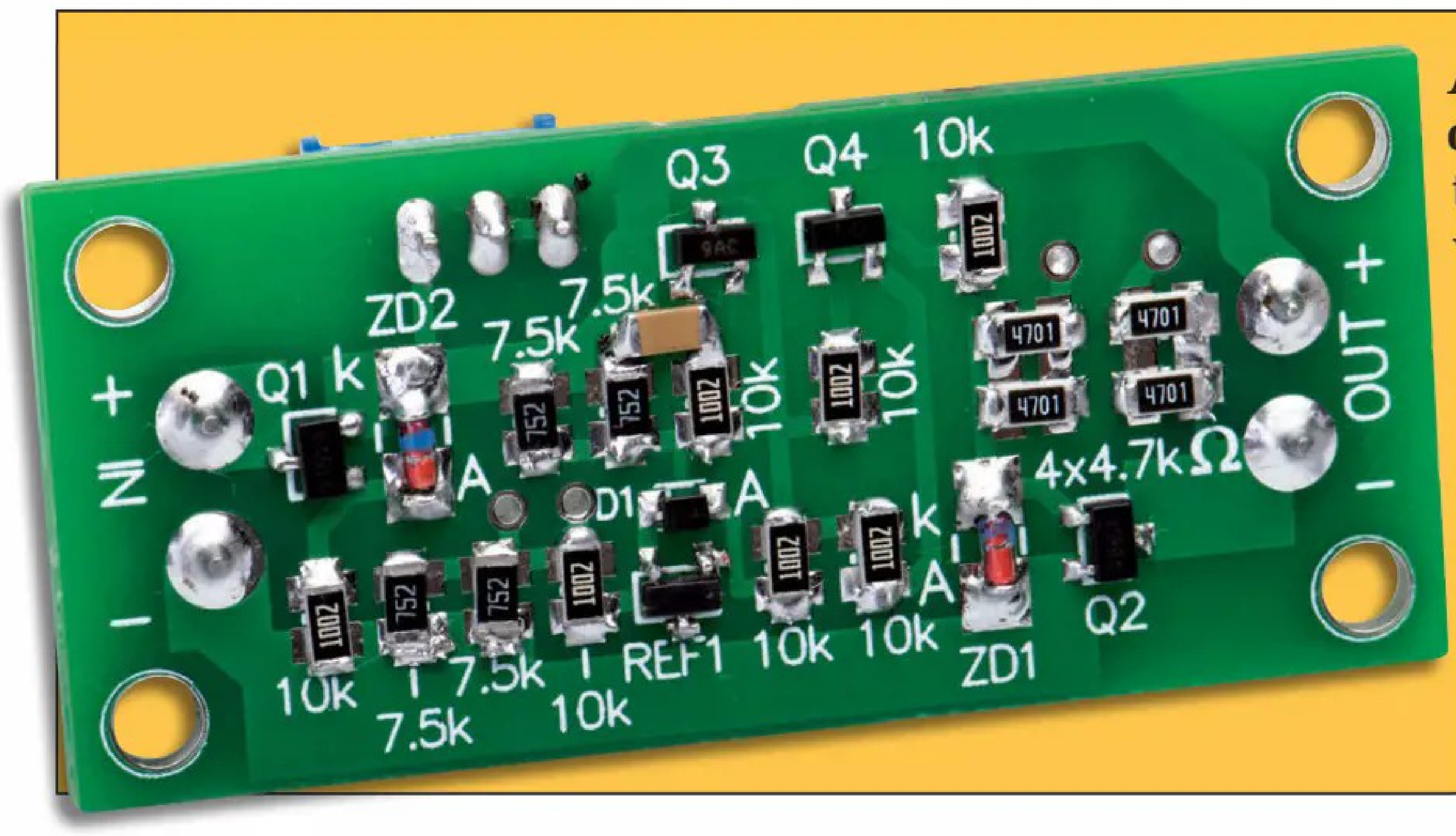
Overvoltage shutdown is indicated when LED2 is on and LED1 is off. When reverse polarity protection is active, both LEDs will be off despite the input power supply being switched on.

TL431 limitations

One thing to note when using the TL431 in the TO-92 through-hole package is that the threshold between switching high or low is closer to 2V than 2.5V. The likely reason is that the reference requires a minimum current to produce the 2.5V reference, which is only available in closed-loop mode. In open-loop mode, the reference is operating further down the threshold knee of the voltage versus current curve.

This threshold also varies with temperature, although provided the temperature does not vary over a wide range, the resulting accuracy will be satisfactory. For more information on using the TL431 in open-loop mode for undervoltage and overvoltage detection, see the Texas Instruments Application Report SLVA987A PDF at www.ti.com/lit/pdf/slva987

The SMD version of the TL431 does not appear to suffer the same problem, as it shows a very sharp voltage-versus-current threshold voltage curve even at very low currents. For this reason, the TH PCB has provision for using


Q3
Q4
10k
ZD2
7.5k
7.5k
Q1
K
IN
+
–
A
D1
A
10k
10k
4x4.7kΩ
OUT
+
–
K
A
10k
7.5k
7.5k
10k
REF1
10k
10k
ZD1
Q2

An enlarged photo of the underside of the Adjustable SMD version of the DC Supply Protector. Compared to the other versions, this one has components mounted on both sides of the PCB.

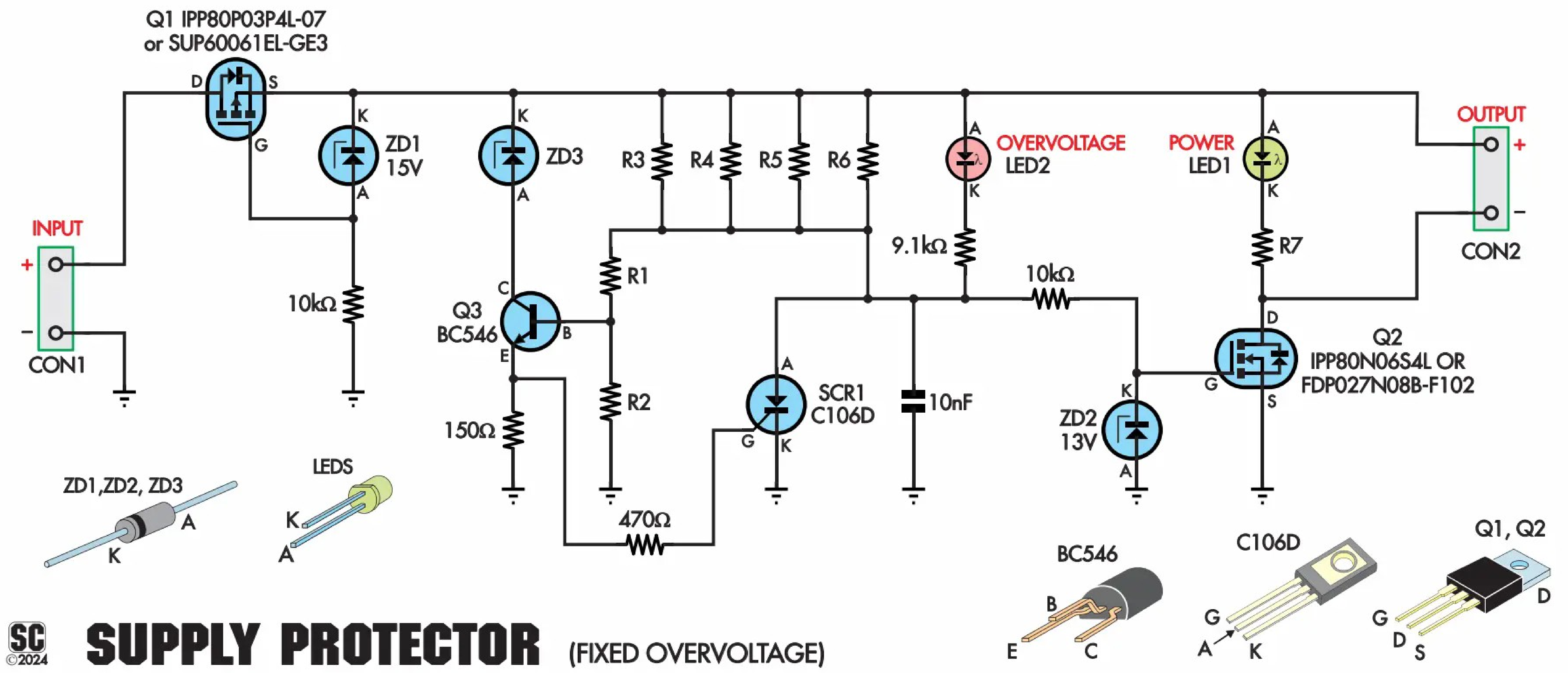

Q1 IPP80P03P4L-07
or SUP60061EL-GE3
D
S
G
K
ZD1
15V
A
K
ZD3
A
R3
R4
R5
R6
A
OVERVOLTAGE
LED2
K
POWER
LED1
A
K
OUTPUT
+
−
CON2
INPUT
+
−
CON1
10kΩ
C
Q3
BC546
B
E
R1
R2
150Ω
9.1kΩ
10kΩ
R7
D
Q2
IPP80N06S4L OR
FDP027N08B-F102
G
S
A
SCR1
C106D
G
K
10nF
K
ZD2
13V
A
470Ω
ZD1,ZD2, ZD3
A
K
LEDS
K
A
BC546
B
E
C
C106D
G
A
K
Q1, Q2
G
D
S
D
SC ©2024
SUPPLY PROTECTOR
(FIXED OVERVOLTAGE)

Fig.3: the fixed overvoltage version requires you to select values for resistors R1-R7 depending on the threshold voltage you want; see Table 1 overleaf. It uses an SCR and zener diode for the over-voltage function rather than a TL431.

the surface-mount version instead of the TO-92 package version.

Fixed overvoltage protector circuit

Fig.3 is the fixed overvoltage protector circuit. This circuit includes reverse polarity protection using P-channel Mosfet Q1 in the same way as Figs.1 & 2. The overvoltage protection also uses N-channel Mosfet Q2, although Q2 is controlled differently in this circuit.

Instead of an adjustable overvoltage threshold controlled by a TL431 shunt reference IC and trimpot, the threshold is set and detected by zener diode ZD3. If the voltage applied to the zener is above the overvoltage threshold and it conducts, silicon-controlled rectifier SCR1 is triggered to switch off Mosfet Q2.

A 10nF capacitor is included across the SCR to prevent it from latching on due to a rapid rise in voltage (dV/dt) as power is initially applied to CON1. Any voltage rise faster than 8V/μs will likely switch the SCR on. The 10nF capacitor slows down the voltage rise.

Mosfet Q2 is normally held on via the gate voltage applied by the 10kΩ gate resistor and paralleled resistors R3-R6. Zener diode ZD2 protects the gate from excessive voltage. With Q2 on, a low-resistance connection exists between the drain and source, connecting the ground of CON1 to the negative side of CON2.

In that case, the power LED (LED1) lights due to the current flow through R7 to the negative side of the supply.

Transistor Q3 is typically switched on by the bias current from the positive supply via resistors R3-R6 and R1. With Q3 on, current can flow through ZD3 at its collector and the 150Ω resistor at its emitter, but only if the supply voltage exceeds ZD3's breakdown voltage.

4mA needs to flow through ZD3 before the voltage across the 150Ω resistor reaches 0.6V, which is just sufficient to trigger SCR1 via its 470Ω gate resistance.

Thus, when the SCR switches on and disconnects the load, the supply voltage is ZD3's rated breakdown voltage plus the 0.6V required across the 150Ω resistor. When SCR1 latches on, there is about 1V between its anode (A) and cathode (k), so Q3 switches off.

With SCR1 on, the low voltage at Q2's gate switches it off, disconnecting the ground supply at CON2. LED1 is now off, while the low voltage across SCR1 causes LED2 to light, with current flowing through the 9.1kΩ resistor to the switched-on SCR.

Note that LED2 will also light when the voltage across ZD2 reaches its breakdown of 13V. As the supply voltage rises, LED2 brightens as more current flows through the LED via the 9.1kΩ resistor and ZD2. Overvoltage shutdown is indicated when LED2 is lit but LED1 is off.

The voltage divider formed with R1 and R2 ensures that Q3's base is well below 0.6V, keeping Q3 off when SCR1 is on. With Q3 off, the gate drive to SCR1 is off, but the SCR remains latched on due to the current flowing through it. Resistors R3 to R6 provide the required 5mA latching and holding current to ensure it stays on in this condition.

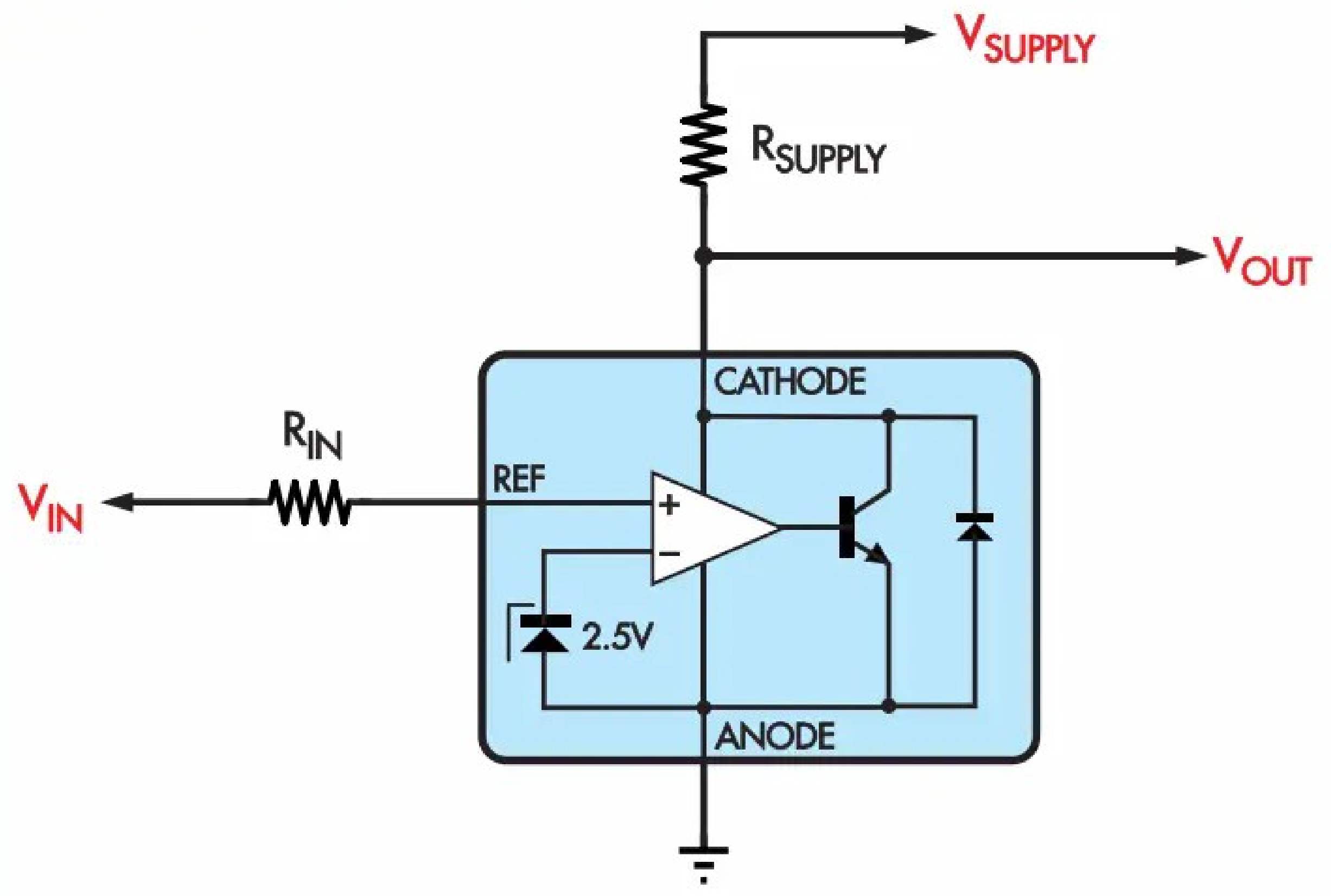

V_SUPPLY
R_SUPPLY
V_OUT
R_IN
V_IN
CATHODE
REF
+
−
2.5V
ANODE

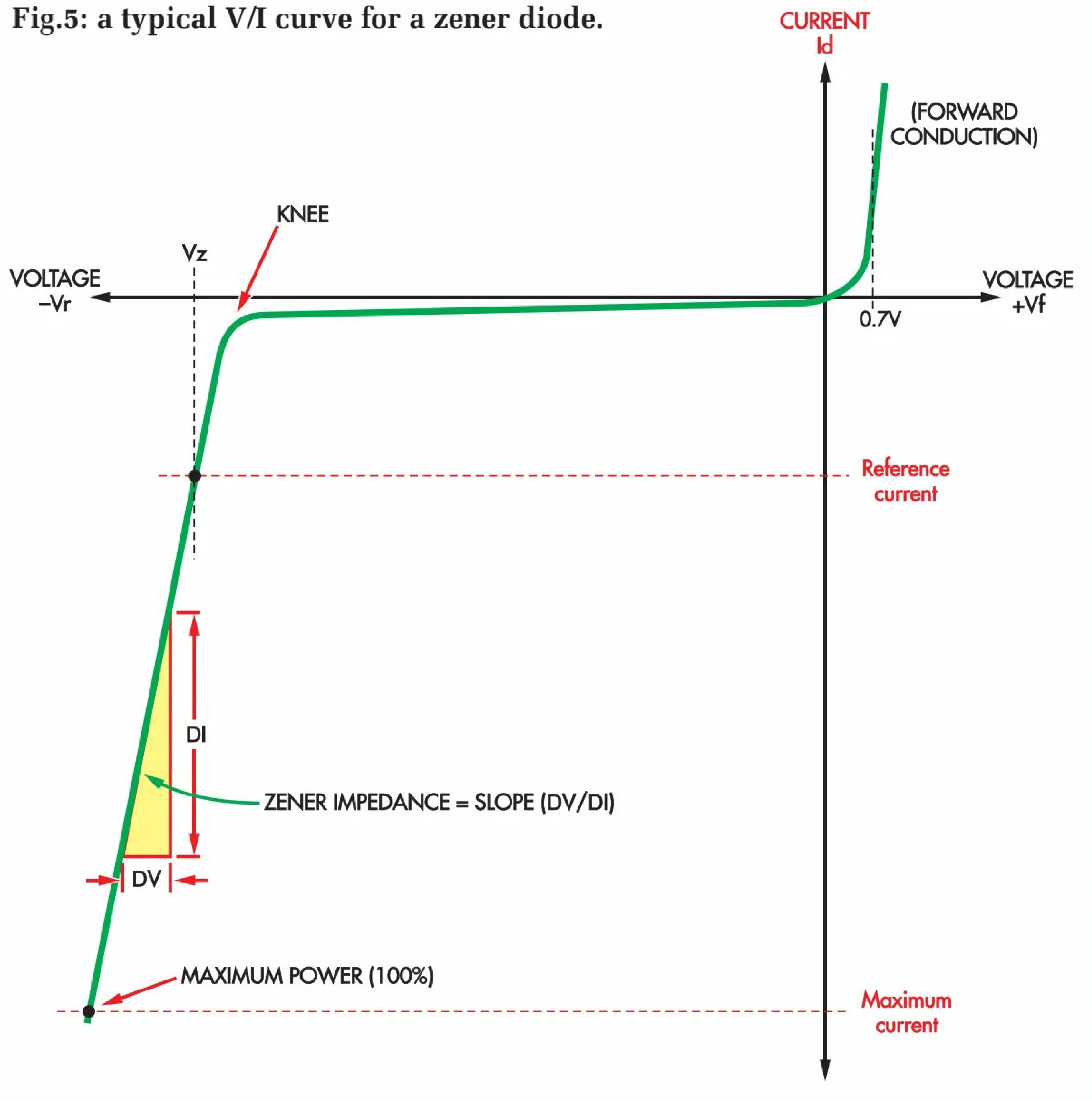
Fig.4: the basic circuitry within a TL431 voltage reference. Usually, the REF terminal is connected to a divider between the anode and cathode (closed-loop mode). Here, we are using it in open-loop mode, as a voltage detector.

Table 1 – resistance values for fixed TH version

ZD3	V_{OVL}	R1	R2	R3	R4	R5	R6	R7
47V	47.7V	130kΩ	13kΩ	18kΩ	18kΩ	×	×	8.2kΩ
43V	43.7V	110kΩ	13kΩ	16kΩ	16kΩ	×	×	6.8kΩ
39V	39.7V	100kΩ	13kΩ	15kΩ	15kΩ	×	×	5.6kΩ
36V	36.7V	91kΩ	13kΩ	16kΩ	13kΩ	×	×	4.3kΩ
30V	30.7V	75kΩ	13kΩ	12kΩ	12kΩ	×	×	3.0kΩ
27V	27.7V	68kΩ	13kΩ	5.6kΩ	×	×	×	2.4kΩ
24V	24.7V	62kΩ	13kΩ	4.7kΩ	×	×	×	2.2kΩ
22V	22.7V	56kΩ	13kΩ	8.2kΩ	10kΩ	×	×	2.0kΩ
20V	20.7V	51kΩ	13kΩ	8.2kΩ	8.2kΩ	×	×	1.8kΩ
16V	16.7V	36kΩ	10kΩ	10kΩ	10kΩ	8.2kΩ	×	1.3kΩ
15V	15.7V	33kΩ	10kΩ	10kΩ	8.2kΩ	8.2kΩ	×	1.3kΩ
13V	13.7V	30kΩ	10kΩ	8.2kΩ	8.2kΩ	6.8kΩ	×	1.2kΩ
12V	12.7V	27kΩ	8.2kΩ	6.8kΩ	6.8kΩ	6.8kΩ	×	1kΩ
11V	11.7V	24kΩ	8.2kΩ	5.6kΩ	6.8kΩ	6.8kΩ	×	1kΩ
10V	10.7V	18kΩ	6.2kΩ	6.8kΩ	5.6kΩ	5.6kΩ	×	910Ω
9.1V	9.8V	15kΩ	4.3kΩ	5.6kΩ	5.6kΩ	5.6kΩ	×	820Ω
8.2V	8.9V	12kΩ	4.3kΩ	4.7kΩ	4.7kΩ	4.7kΩ	×	750Ω
7.5V	8.2V	7.5kΩ	2.4kΩ	5.6kΩ	5.6kΩ	5.6kΩ	5.6kΩ	620Ω
6.8V	7.5V	3.6kΩ	1.2kΩ	4.7kΩ	4.7kΩ	5.6kΩ	5.6kΩ	560Ω

White = ½W, yellow = 1W, × = not fitted

Fig.5: a typical V/I curve for a zener diode.



If you are wondering why we need Q3 instead of ZD3 connecting directly in series with the 150Ω resistor, it is because ZD3 could be damaged by excessive current as the supply voltage rises well above its breakdown voltage. For example, if ZD3 is a 12V zener diode, it will conduct 4mA when the supply is at 12.6V but 186mA at 40V. In that case, it would be running well above its power rating.

Additionally, the 150Ω resistor would be dissipating just over 5W. Having transistor Q3 means that all this current stops once the overvoltage threshold is reached, preventing high dissipation in ZD3 and the 150Ω resistor.

Zener diode biasing

The 150Ω resistor could be increased in value, but that would mean that the overvoltage threshold would occur at a much lower voltage than the zener diode breakdown voltage. This would be less consistent than using the zener at the steeper region of its conduction curve.

Fig.5 shows a typical zener diode V/I curve. In the forward direction (current flowing from anode to cathode), it acts like a regular diode, conducting current with 0.6-0.7V voltage across it. In the reverse direction, the zener initially acts like a diode, blocking current with minimal leakage current.

However, beyond a certain voltage, the ‘leakage’ current increases and then it begins conducting significant reverse current. This is the reverse breakdown mode, which provides a relatively steep VI curve beyond the knee region.

Each zener diode is characterised at a particular current for its zener voltage. If the zener diode is operated at a current much less than that, the voltage across it will also be lower. For our circuit, we want the zener diode operating more in the linear region, where the V/I curve is steep, rather than in the knee region and preferably near to the reference current for the zener.

The recommended BZX79Cxx series of zener diodes for our circuit are characterised for a 5mA reference current between 2.4V to 24V, or 2mA above that. The 4mA current for the zener diode in our circuit is a reasonable compromise between those.

Resistance value calculations

The resistance values required for resistors R1 to R7 depend on the overload voltage (V_{OVL}), the maximum input voltage (V_{MAX}) and the latching and holding current for SCR1. Resistor power ratings, LED currents, transistor Q3's base current and ZD3's current need to be considered.

Table 1 shows the resistor values and wattage ratings for various overvoltage thresholds and a 50V maximum applied input voltage. A panel describes the calculations used to formulate that table in more detail.

SMD adjustable version

The SMD adjustable version is built using a double-sided plated-through PCB coded 08106241 that measures 51 × 23mm. As shown in the overlay diagrams (Fig.6), all the SMD parts except the two LEDs mount on one side of the PCB, with the through-hole parts such as the two screw terminals and trimpot on the other side.

Begin by soldering the SMDs. That can be done by soldering one lead of the component first, holding it in place with tweezers. Once it is aligned and positioned correctly (by remelting the solder if necessary), the remaining lead(s) are then soldered. A good light and a magnifying glass are very useful for this task.

You will need to identify the parts first. The resistors are marked with three or four digit codes as shown in the parts list. The 100nF capacitor will not be marked. The smaller semiconductors in SOT-23 packages also have component markings, as per the parts list (although they can vary).

The 10V zener diodes are cylindrical with blue markings at the cathode (k) end. Diode D1 also has a polarity stripe at the cathode end.

Note that the TL431 can have alternative pinouts, with the standard pinout having the cathode at left and reference at right when the anode pin is at the top. The mirrored pinout has the cathode and reference pins transposed. We have provided for both orientations on the PCB by having a 6-pad footprint instead of just the three required for one pinout of the device.

The TL431 must be orientated according to the pinout of the device used. We have marked the pins on the PCB overlay showing the anode, cathode and reference pads. The parts supplied in our kit should be the mirror pin version. The way to check this is to use a multimeter on its diode test across the cathode and reference pins.

You should get a reading of one diode drop (around 0.7V) when the red probe is on the REF pin and the black probe on the cathode pin. You can then orientate it correctly on the PCB and solder it in place. While doing that, be careful not to let solder bridge the used and unused pads. If that happens, use a bit of solder wicking braid can be used to remove the excess solder (adding flux paste will make it easier).

When installing the diodes, make sure these are orientated correctly. The anode (A) and cathode (k) orientations are marked on the PCB overlay.

Once all the surface mount parts have been soldered in place on the one side, flip it over and fit the LEDs, taking care to place each with its correct orientation (checked as mentioned earlier) and in the correct position with regard to colour. These are green for power and red for overvoltage, although you are free to customise the colours if desired.

Ideally, the surface mount LEDs should be tested using the diode test mode of a multimeter. With the red probe on the anode and black lead on the cathode, the LED should light and show its colour. We used green for power and red for overload. There is often a stripe or dot on the cathode but we have seen LEDs with a marking on the anode, so it's better to test them.

The trimpot is installed with the top screw adjustment orientated as shown. This provides an increasing overvoltage threshold with clockwise rotation. The two screw terminals should be mounted with the wire entry toward the outside of the PCB at each end.

TH adjustable version

The through-hole adjustable version is built on a double-sided plated-through PCB that's 08106242 and measures 70.5 × 35.5mm. Refer to Fig.7, the PCB overlay diagram, during the assembly process.

If you are using the SMD TL431 version, install it first, but be careful as they can have alternative pinouts with the reference and cathode transposed. See the instructions a few paragraphs above on determining which pinout you have, aligning it with the PCB markings and soldering it.

The zener diodes and diode D1 can be fitted next. ZD1 is a 15V type, while ZD2 is rated at 13V. These each need to be orientated as shown in Fig.7,

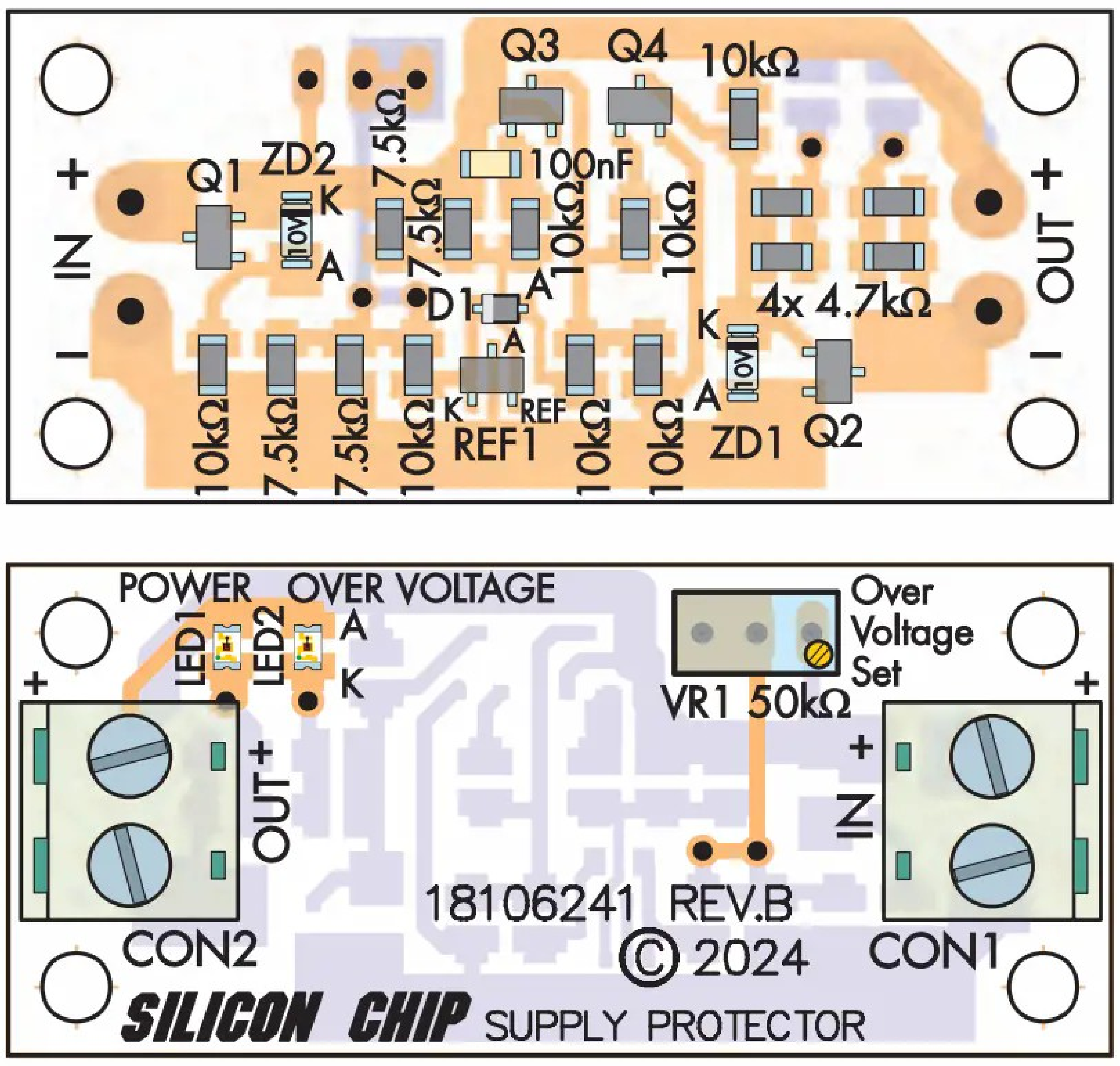

Q3
Q4
10kΩ
Q1
ZD2
7.5kΩ
100nF
10V
K
A
7.5kΩ
10kΩ
10kΩ
+
IN
–
OUT
D1
4x 4.7kΩ
10kΩ
7.5kΩ
7.5kΩ
10kΩ
REF1
REF
10kΩ
10kΩ
ZD1
Q2
POWER
OVER VOLTAGE
LED1
LED2
A
K
Over Voltage Set
VR1 50kΩ
+
OUT
IN
18106241 REV.B
© 2024
CON2
CON1
SILICON CHIP SUPPLY PROTECTOR

Fig.6 (left): the overlay diagrams for the SMD adjustable version of the Supply Protector (shown at 150% actual size).

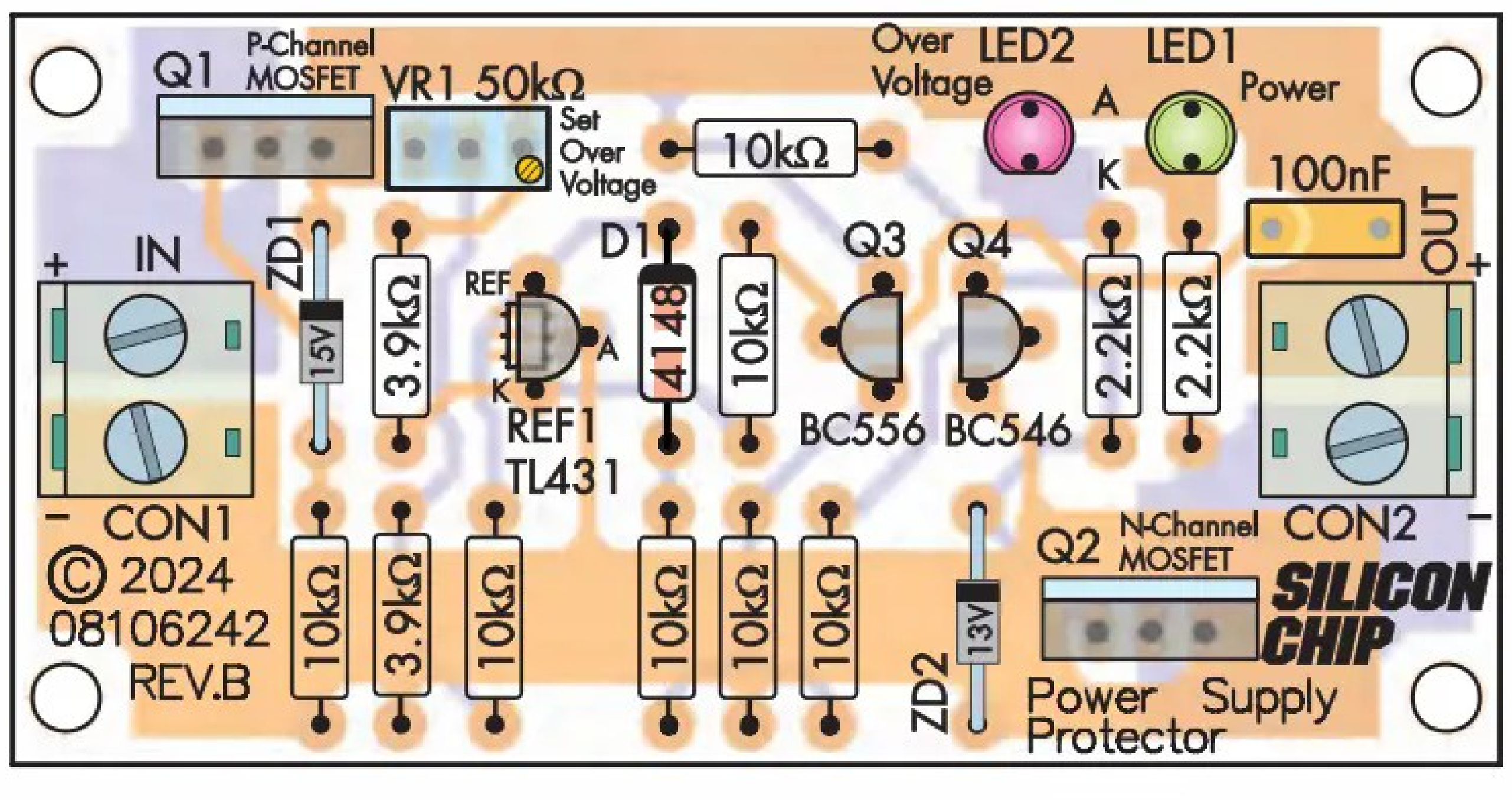

Q1
P-Channel MOSFET
VR1 50kΩ
Set Over Voltage
10kΩ
Over Voltage
LED2
LED1
Power
A
K
100nF
OUT
IN
ZD1
15V
3.9kΩ
REF
D1
4148
10kΩ
Q3
Q4
2.2kΩ
2.2kΩ
REF1
TL431
BC556
BC546
CON1
© 2024
08106242
REV.B
10kΩ
3.9kΩ
10kΩ
10kΩ
10kΩ
10kΩ
ZD2
13V
Q2
N-Channel MOSFET
CON2
SILICON CHIP
Power Supply Protector

Fig.7 (upper right): the PCB overlay diagram for the through-hole adjustable version.

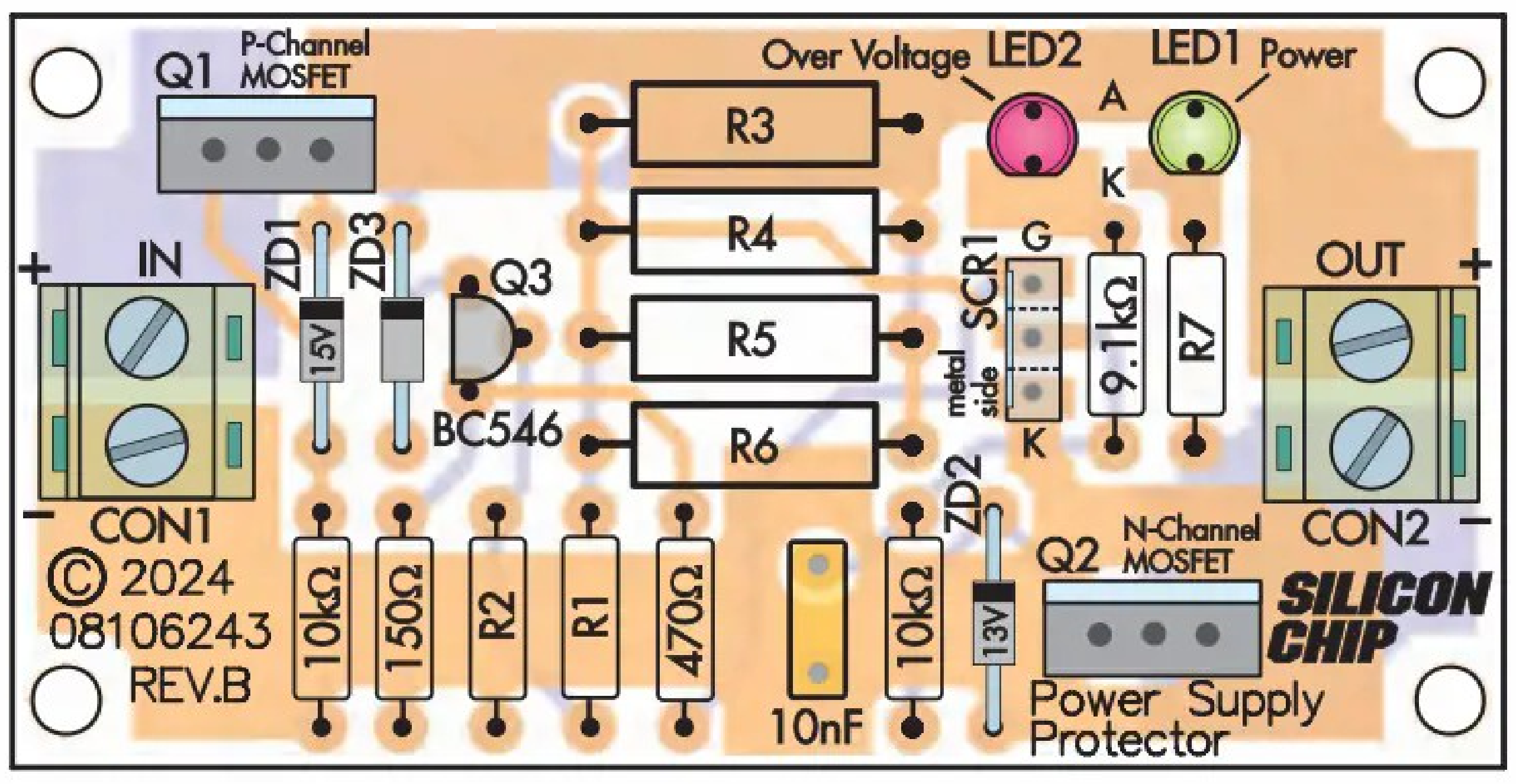

Q1
P-Channel MOSFET
Over Voltage
LED2
LED1
Power
R3
A
K
IN
ZD1
ZD3
Q3
R4
G
OUT
15V
R5
SCR1
9.1kΩ
R7
BC546
R6
metal side
K
CON1
© 2024
08106243
REV.B
ZD2
10kΩ
150Ω
R2
R1
470Ω
10nF
10kΩ
13V
Q2
N-Channel MOSFET
CON2
SILICON CHIP
Power Supply Protector

Fig.8 (lower right): the PCB overlay diagram for the through-hole fixed overvoltage version.

with the cathode band toward the top.

The resistors can be mounted next; check each value with a multimeter to be sure the correct value is installed in each place.

The two LEDs are installed with the tops of their domes about 12mm above the top of the PCB. Check which colour the diode is before installing it, using the diode test mode on a multimeter if the lenses aren't tinted. We used a green LED for power (LED1) and red for overvolage (LED2). In each case, the longer lead is the anode.

Next, fit transistors Q1-Q4, being careful that each is placed in the correct position (check their part codes against Fig.7 and the PCB overlay). If using the TO-92 package version of the TL431 (REF1), you can also fit it now. Follow by mounting the 100nF capacitor.

The trimpot is installed with the screw adjustment orientated as shown, providing an increasing overvoltage threshold with clockwise rotation. The two screw terminals are mounted with the wire entry toward the outside of the PCB at each end.

TH fixed overvoltage version

The through-hole fixed overvoltage version is built on a double-sided, plated-through PCB coded 08106243 that measures 70.5 × 35.5mm. The PCB overlay diagram for this version is Fig.8.

First, the values for resistors R1-R7 need to be selected using Table 1, based on the required overvoltage threshold. The required voltage rating for ZD3 is also listed in that table. Note that resistors R3-R6 may need to be 1W types (if shown in yellow in Table 1), and not all four of these resistors are necessarily used for all possible threshold voltages.

The zener diodes and diode D1 can be fitted now. ZD1 is rated at 15V, ZD2 is a 13V type, while ZD3 is as per Table 1. These each need to be orientated as shown in Fig.8, with the cathode band toward the top.

The resistors can be mounted next; check each value with a multimeter to be sure the correct value is used in each location.

The two LEDs are installed with the tops of their domes about 12mm above the top of the PCB. Check which colour the diode is before installing it, using the diode test on a multimeter if the lenses aren't tinted. We used green

Resistance value calculations

Table 1 shows the required resistance values and power ratings for the Fixed Protector for overvoltage thresholds from 7.5V to 47.7V with a maximum input voltage of 50V. There are no satisfactory resistance values to meet all requirements for overvoltage thresholds below 7.5V, so if you require a threshold that low, build one of the other versions.

R3 to R6 calculations

The total resistance for R3 to R6 is calculated first. This resistance provides current for SCR1 and the base of transistor Q3 via R1. Up to four resistors can be paralleled for a sufficient power rating and to achieve the required resistance.

The latching and holding current required for SCR1 to remain in conduction is 5mA. This satisfies the worst-case latching current and the worst-case holding current at 25°C.

The total resistance, R, required is the overload voltage threshold (V_{OVL}) minus one volt (the SCR anode-to-cathode on-voltage), divided by 5mA, ie, $R = (V_{OVL} - 1V) \div 5mA$.

The total power rating required is the maximum operating voltage for the circuit (eg, 50V) minus 1V squared and then divided by the resistance, ie, $(V_{MAX} - 1V)^2 \div R$.

The required power rating can be reduced by spreading it between two to four resistors in parallel. If all those resistors have the same value, they share the dissipation equally. If different, each resistor will need to be assessed for its share of the dissipation.

R1 & R2 value calculations

Resistor R1 drives the base of Q3, which must saturate when conducting 4mA. This 4mA is the current that flows through ZD3, Q3 and the 150Ω resistor at the overvoltage threshold. We drive Q3's base with 250µA ($^1/_{16}$th the collector current) or more to ensure Q3 goes into saturation.

Resistor R2, between the base of Q3 and ground, is necessary since it reduces the base voltage to less than 0.3V due to divider action with R1 once SCR1 is latched. Typically, SCR1 will have about 1V across, so provided that R1 is at least triple R2's value, that will be reduced to 250mV or less. That prevents Q3 from conducting through ZD3 once overvoltage has been detected and SCR1 latches on.

For overvoltage settings of 20.7V and above, we set R2 so 100µA flows through it at the overload voltage threshold. At this threshold, there will be 0.66V between the base and emitter of Q3 and 0.6V at the emitter of Q3, giving a total of 1.26V across R2. For an approximate 100µA current, R2 needs to be 12.6kΩ (13kΩ is the closest E24 value). 13kΩ gives 96.9µA, close enough to 100µA.

When calculating the value for R1, this 100µA needs to be included since this bypasses the current from Q3's base. So, instead of R1 supplying 250µA to Q3's base, it needs to supply 350µA in total.

R1 is calculated as the overload voltage threshold (V_{OVL}) minus the 1.26V at Q3's base, divided by 350µA. Since R1 is in series with the R3-R6, the parallel value of R3-R6 is then subtracted from this to get R1's value, ie, $R1 = (V_{OVL} - 1.26V) \div 350\mu A - (R3 \parallel R4 \parallel R5 \parallel R6)$.

If the calculated value for R1 is less than three times the value of R2, the current through R2 needs to be increased and the equations reworked. For example, to get 200µA through R2, $R2 = 1.26V \div 200\mu A = 6.3k\Omega$ (use 6.2kΩ). Then $R1 = (V_{OVL} - 1.26V) \div 450\mu A$, where 450µA is the 200µA R2 current plus the 250µA required for Q3's base.

LED current

LED1 switches off above the overvoltage threshold, so the maximum LED current will occur with the supply at the overvoltage setting. Assuming 10mA is a suitable maximum current, the value for R7 is the overload voltage minus the 2V across the LED, divided by 10mA, ie, $R7 = (V_{OVL} - 2V) \div 10mA$.

The power rating for R7 also needs to be considered, so its value needs to be greater than $(V_{OVL} - 2V)^2 \div 250mW$, where 250mW is a conservative derating for a 500mW resistor. If this calculation gives a higher value than the above, the maximum LED current will be below 10mA to avoid overheating the current-limiting resistor.

The overvoltage LED (LED2) series resistor value is calculated similarly; only this time, the maximum input supply voltage is used in the calculation. That's because LED2 will light from the overvoltage threshold to the maximum input supply voltage, V_{MAX}. So the calculation is V_{MAX} minus the voltage across LED2 and SCR1, divided by 10mA, ie, $R = (V_{MAX} - 3V) \div 10mA$.

Similarly, the minimum value, considering the resistor power rating, is $(V_{MAX} - 3V)^2 \div 250mW$. We selected a 9.1kΩ $^1/_2$W resistor for a V_{MAX} of 50V.

for the power LED (LED1) and red for overvoltage (LED2). In each case, the longer lead is the anode.

Be sure when mounting Q1 to Q3 that each is placed in the correct position and orientation. The SCR goes in with the metal tab side towards R3-R6.

The trimpot should be installed with the screw adjustment orientated as shown, providing an increasing overvoltage threshold with clockwise rotation. The two screw terminals are mounted with the wire entry toward the outside of the PCB at each end.

Testing

If you have an adjustable power supply, you can apply power to the input and check that the power LED lights and the overvoltage switch-off function operates at the desired voltage. This is preset with the fixed version or can be changed using VR1 for the adjustable versions.

Once the overvoltage threshold has been reached, the power LED goes off and the overvoltage LED lights up.The supply will need to be switched off or significantly reduced before power is restored to the output.

Also remember that the overvoltage LED may light once the supply voltage exceeds ZD2's breakdown voltage. Overvoltage shutdown is indicated when the power LED (LED1) is off and the overvoltage LED (LED2) is lit, but not when both LEDs are alight.

For the adjustable versions, you can set the overvoltage threshold approximately by measuring the resistance across VR1 when the power is off. Divide the VR1 resistance by 3.9kΩ, add one, then multiply by 2V if you used a TO-92 TL431 or 2.5V if you used the SMD version. The formula is $V_{OVL} = (R(VR1) \div 3.75k\Omega + 1) \times V_{REF}$. That will tell you roughly what voltage it will cut out at, within about 1V.

For the reverse calculation, to determine what resistance you need across VR1 for an approximate voltage threshold, divide the desired threshold by 2V (TO-92 TL431) or 2.5V (SMD TL431), then subtract one and multiply by 3.9kΩ (3.75kΩ for the SMD version) The formula is $R(VR1) = (V_{OVL} \div V_{REF} - 1) \times 3.9k\Omega$.

To set it more accurately, you will need an adjustable power supply or make a basic one using a wirewound 1kΩ potentiometer connected across a fixed supply (but be careful not to exceed its power rating).

Parts List – DC Supply Protectors

Common between all versions

2 2-way PCB mount screw terminals with 5mm or 5.08mm spacing (CON1, CON2)

SMD Adjustable Version

1 double-sided, plated-through PCB coded 08106241, 51 × 23mm
1 100nF 50V X7R ceramic capacitor, SMD 3216/1206 size
1 50kΩ multiturn top-adjust trimpot, Bourns 3296W style (VR1)

Semiconductors
1 AO3401(A) 30V 4A P-channel logic-level Mosfet, SOT-23 (Q1; marking: X15V)
1 AO3400 30V 5.8A N-channel logic-level Mosfet, SOT-23 (Q2; marking: XORB)
1 BC856C 65V 100mA PNP transistor, SOT-23 (Q3; marking: 9AC)
1 BC846C 65V 100mA NPN transistor, SOT-23 (Q4; marking: 1C)
1 TL431 adjustable shunt voltage reference, SOT-23 (REF1; marking: 431) ●
1 1N4148WS 75V 150mA switching diode, SOD-323 (D1)
2 BZV55-C10 10V 500mW zener diodes, SOD-80C (ZD1, ZD2)
1 green SMD LED, M3216/1206 size (LED1)
1 red SMD LED, M3216/1206 (LED2)

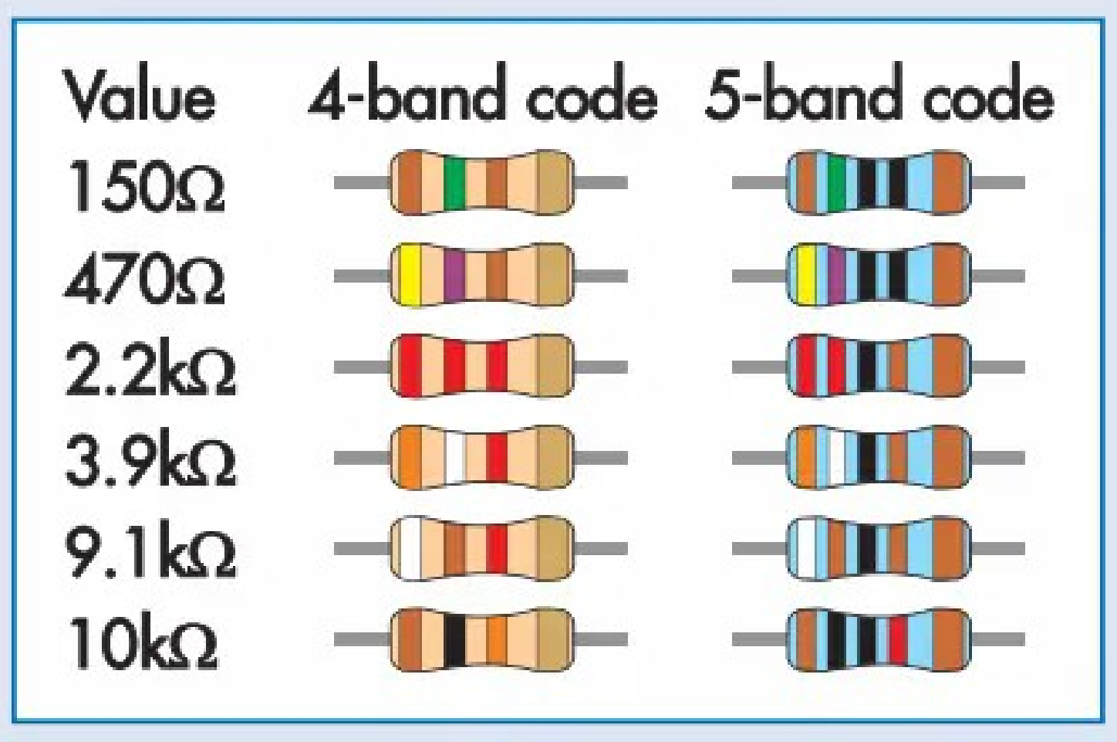
Resistors *(all M3216/1206 size ¼W 1% SMD)*
7 10kΩ (code 1002 or 103)
4 7.5kΩ (code 7501 or 752)
4 4.7kΩ (code 4701 or 472)

Through-Hole Adjustable Version

1 double-sided, plated-through PCB coded 08106242, 70.5 × 35.5mm
1 100nF 63V/100V MKT polyester capacitor
1 50kΩ multiturn top-adjust trimpot, Bourns 3296W style (VR1)

Semiconductors
1 IPP80P03P4L-07 or SUP60061EL-GE3 P-channel logic-level Mosfet, TO-220 (Q1)
1 IPP80N06S4L or FDP027N08B-F102 N-channel logic level Mosfet, TO-220 (Q2)
1 BC556 65V 100mA PNP transistor, TO-92 (Q3)
1 BC546 65V 100mA NPN transistor, TO-92 (Q4)
1 TL431 adjustable shunt voltage reference, TO-92 (REF1) ***OR***
1 TL431 adjustable shunt voltage reference, SOT-23 (REF1; marking: 431) ●
1 1N4148 75V 200mA signal diode (D1)
1 15V 500mW or 1W zener diode (ZD1)
1 13V 500mW or 1W zener diode (ZD2)
1 3mm green LED (LED1)
1 3mm red LED (LED2)

Resistors *(all ½W metal film, 1%)*
7 10kΩ
2 3.9kΩ
2 2.2kΩ



Value	4-band code	5-band code
150Ω		
470Ω		
2.2kΩ		
3.9kΩ		
9.1kΩ		
10kΩ		

Through-Hole Fixed Overvoltage Version

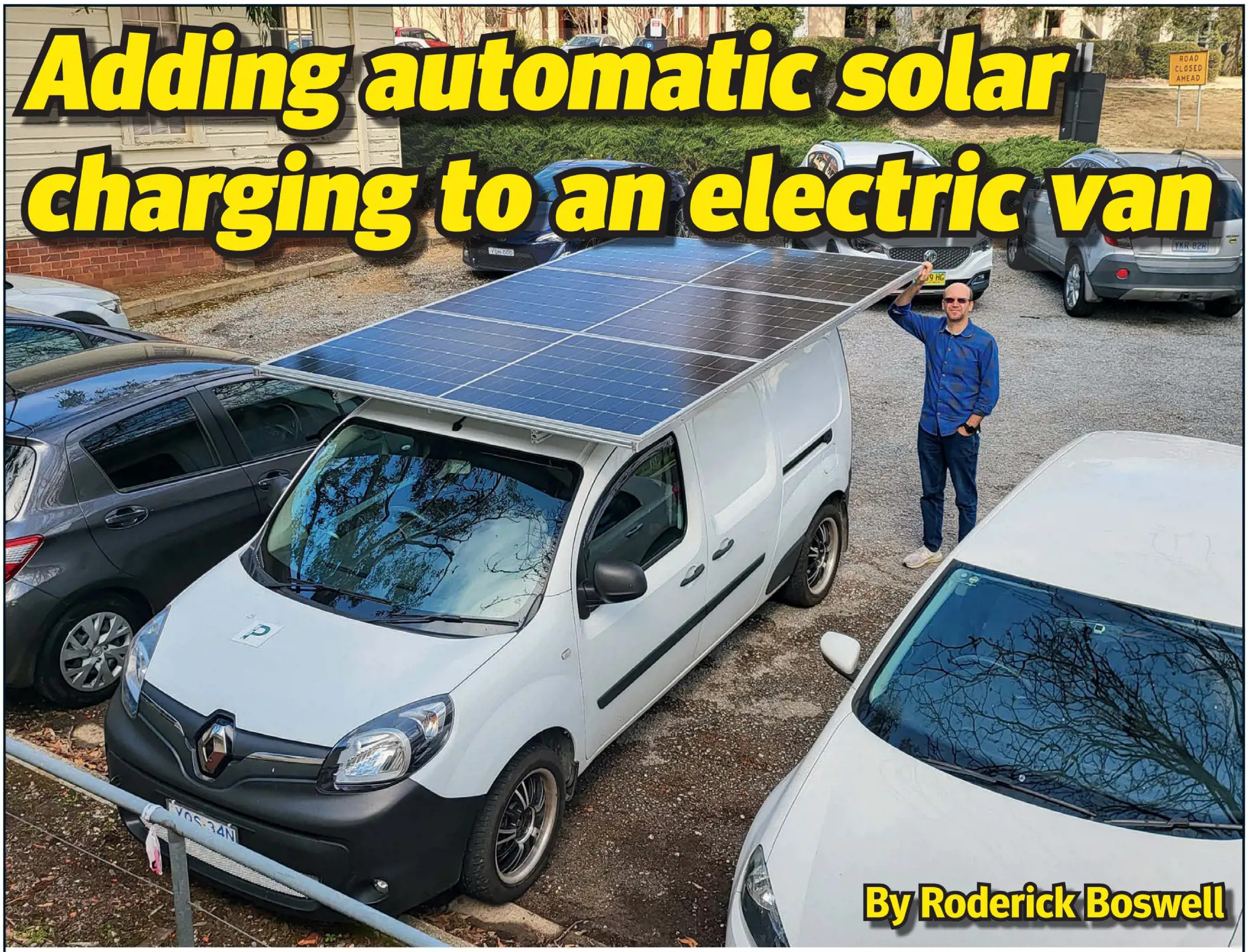
1 double-sided, plated-through PCB coded 08106243, 70.5 × 35.5mm
1 10nF 63V/100V MKT polyester capacitor

Semiconductors
1 IPP80P03P4L-07 or SUP60061EL-GE3 P-channel logic-level Mosfet, TO-220 (Q1)
1 IPP80N06S4L or FDP027N08B-F102 N-channel logic level Mosfet, TO-220 (Q2)
1 BC546 65V 100mA NPN transistor, TO-92 (Q3)
1 C106B 200V or C106D 400V 4A SCR, TO-126/TO-225AA (SCR1)
1 15V 500mW or 1W zener diode (ZD1)
1 13V 500mW or 1W zener diode (ZD2)
1 BZX79Cxx 500mW (2mA or 5mA reference current) zener diode (ZD3)
[See Table 1 for voltage rating]
1 3mm green LED (LED1)
1 3mm red LED (LED2)

Resistors *(all ½W metal film, 1%)*
2 10kΩ
1 9.1kΩ
1 470Ω
1 150Ω
R1-R7: see Table 1

● *TL431QDBZR, TL431FDT or TL431SDT have the standard pinout; TL431MFDT or TL431MSDT have the mirrored pinout*

Adding automatic solar charging to an electric van



By Roderick Boswell

How far can an EV travel without having to visit a charger? We added solar panels to the roof of a Renault Kangoo ZE van, plus an onboard inverter. This gives us up to 18,000km (11,000 miles) a year of driving at no further cost!

Having built the solar van, we've achieved up to 50km (30 miles) of driving per day using just the solar panels. Multiply that by the number of days in a year to get the 18,000km (11,000 miles) figure, although that assumes nice sunny weather year-round, which is perhaps a little unrealistic.

Still, sitting in the van watching the onboard batteries charge at 50A for the first time, it certainly was pleasing to realise that it was working as intended. There are surprisingly few reports of this having been done, so we thought we would create a company, "Solely Solar", based on the concepts of autonomy and freedom.

In this article, I will describe how the decisions were made, what we purchased, how we configured and tested it, the integration of the solar system into the van and the on-road tests.

The solar panels

Photovoltaic (PV) solar panels have been slowly improving over the past few years. It is possible to purchase single crystal silicon arrays with passivated emitter rear cells (PERC), which were invented by a team at UNSW in Sydney. They are cut in half to reduce the resistance and hence losses.

These cells have an efficiency of around 22%, so with full sun delivering 1kW per square metre, you can obtain 220W from a 1m² panel.

So, off I went to eBay to check prices; I found a real Aladdin's cave of solar treasures. Having purchased a few, I quickly discovered that the power they could produce was often overstated by as much as 100%.

Unless you like opening protracted disputes with eBay (which I did to see how the system works; it does, sort of), be aware that the only reliable indicator of the potential power of the panel is its area. I learned this by spending money and testing the product, an easy task with a multimeter that can measure up to 20A.

The two main parameters to measure are the open circuit voltage (V_{OC}), which increases with the area of the panel, and the short-circuit current (I_{SC}), which manufacturers try to keep as low as possible to reduce Joule heating (I^2R). For example, I tested a 2m² solar panel with a V_{OC} of around 50V and an I_{SC} of around 10A.

Of course, multiplying those two

figures won't tell you exactly how much the panel will produce since they are measured under different conditions. Still, it gives you a way to estimate the power and compare different panels.

We decided to use Longi 510W panels that measured 2093 × 1134mm and weighed 25.3kg since they just fitted onto the roof of the Kangoo. Interestingly, some tests showed around 550W being produced per panel.

There is an efficiency temperature coefficient of -0.25%/°C, with the stated performance being at 25°C. So, on a cold morning, with a temperature around 0°C, the panels will be 5% more efficient. Conversely, of course, during the afternoon in summer, the air temperature may be 40°C, and the panels will be so hot you cannot touch them, leading to a performance reduction of up to 10%.

The van

There are currently several very expensive electric vans on the market. Still, a couple of years ago, the only real option was the Renault Kangoo Zero Emission, although BYD slipped around 50 T3 electric vans into Australia as they were mucking about with distributors.

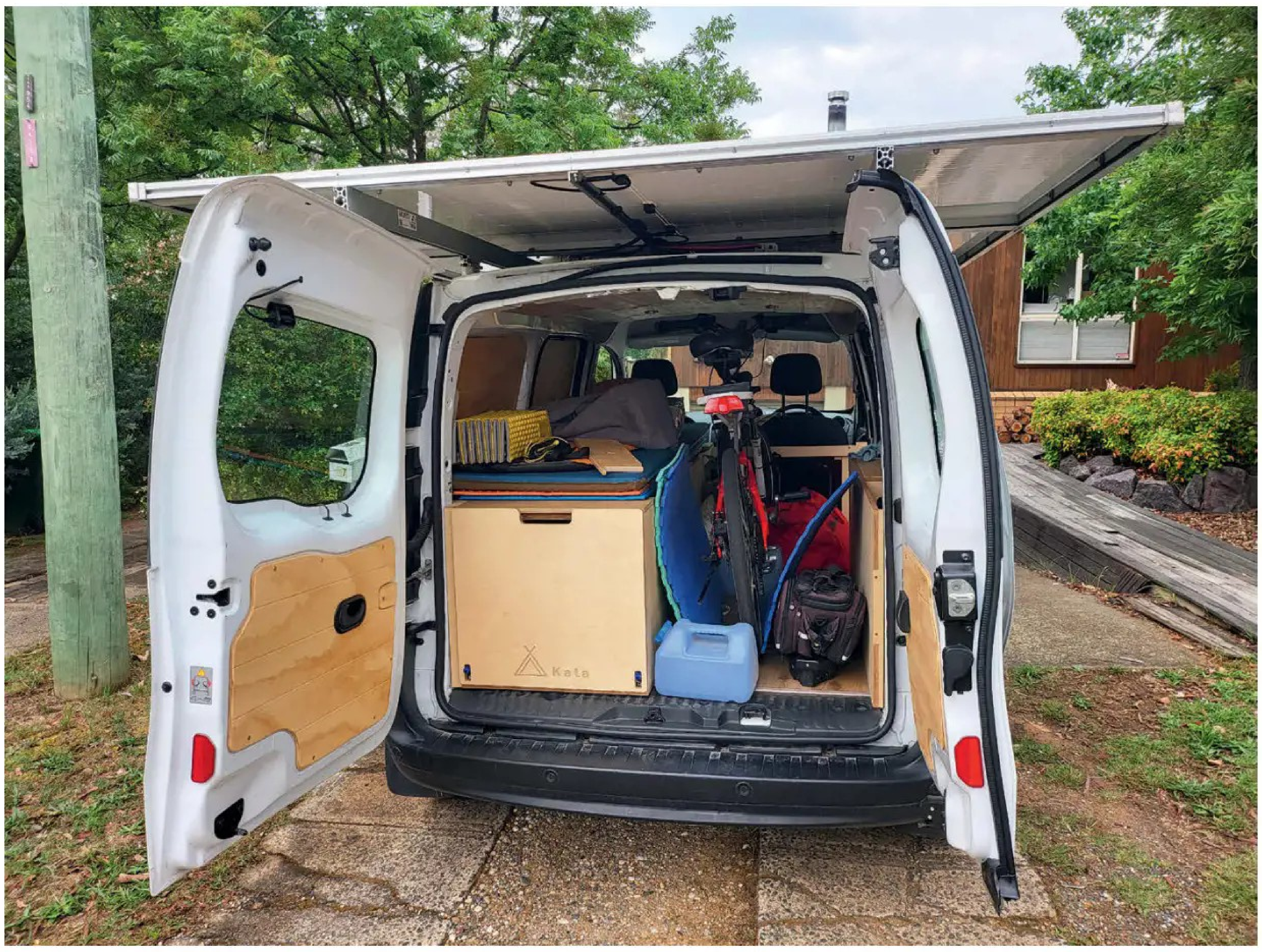
I decided on a 2019 Kangoo ZE that had been used to drive from the Blue Mountains to Sydney every day and back, which had travelled around 85,000km (53,000 miles). The Kangoo has a Mennekes Type 2 7kW onboard charger that requires a Type 2 to Type 2 cable, or a destination charging cable with a Type 2 on one end and a regular 10A 230V mains plug at the other.

Two of us would have to drive to Sydney from Canberra, pick the van up, and drive it back. Since the top of the CCS (Combined Charging System) plug is a Mennekes Type 2 plug, we purchased a Type 2 to Type 2 cable. We made an unpleasant discovery when we stopped at a commercial charging station at Sutton Forest on the way back to Canberra. The top Mennekes socket of the CCS charging station was not connected!

As night was about to fall, we swiftly returned to Canberra in the other car, leaving the Kangoo in the parking area adjacent to the chargers. Rats!

After some research, we found that the commercial CCS charger providers wanted a fast turnover so their chargers only provided DC fast charging. The poor old Mennekes is generally limited to 7kW, resulting in a wait of several hours. The company supplying the chargers evidently didn't want anyone sitting on their charger for that long, so they removed the Type 2 option.

The inside of the Renault Kangoo ZE van with some basic wiring for the solar panels.



The onboard inverter and the other electronics required for the solar panels are stored in the large timber cabinet on the side that doubles as a kitchen.

The following morning, we returned much wiser and drove the Kangoo to a local winery that had a couple of Type 2 chargers. We plugged in and then discovered that you have to download the charger supplier's app on your phone to arrange payment before you can start charging. After doing that, it was finally charging, and we had four hours to kill.

We had lunch at the winery and drove around the area, which was really quite beautiful, and got back to discover that a watched charger never boils. Eventually, we were back on the road again, popping into the Goulburn Workers' Club later for supper and another couple of hours of Type 2 charging. That got us home.

At home, we used the cable with the Type 2 connector on one end and 3-pin mains plug on the other to recharge the van overnight.

The dash instruments show the instantaneous kWh/100km figure, estimated range, distance travelled and instantaneous power usage. It also has a 'fuel gauge' that correlates more-or-less with the battery state of charge (SoC).

At ⅛ SoC remaining, you touch the red line and a warning light suggests you look for a charger, as there is only about 30km (19 miles) of range remaining. A double red line follows at 1/16, and another light appears that the manual explains is to warn that you are about to go into 'limp home' mode. I checked this scheme out, down to 1/16th full, and all worked as expected.

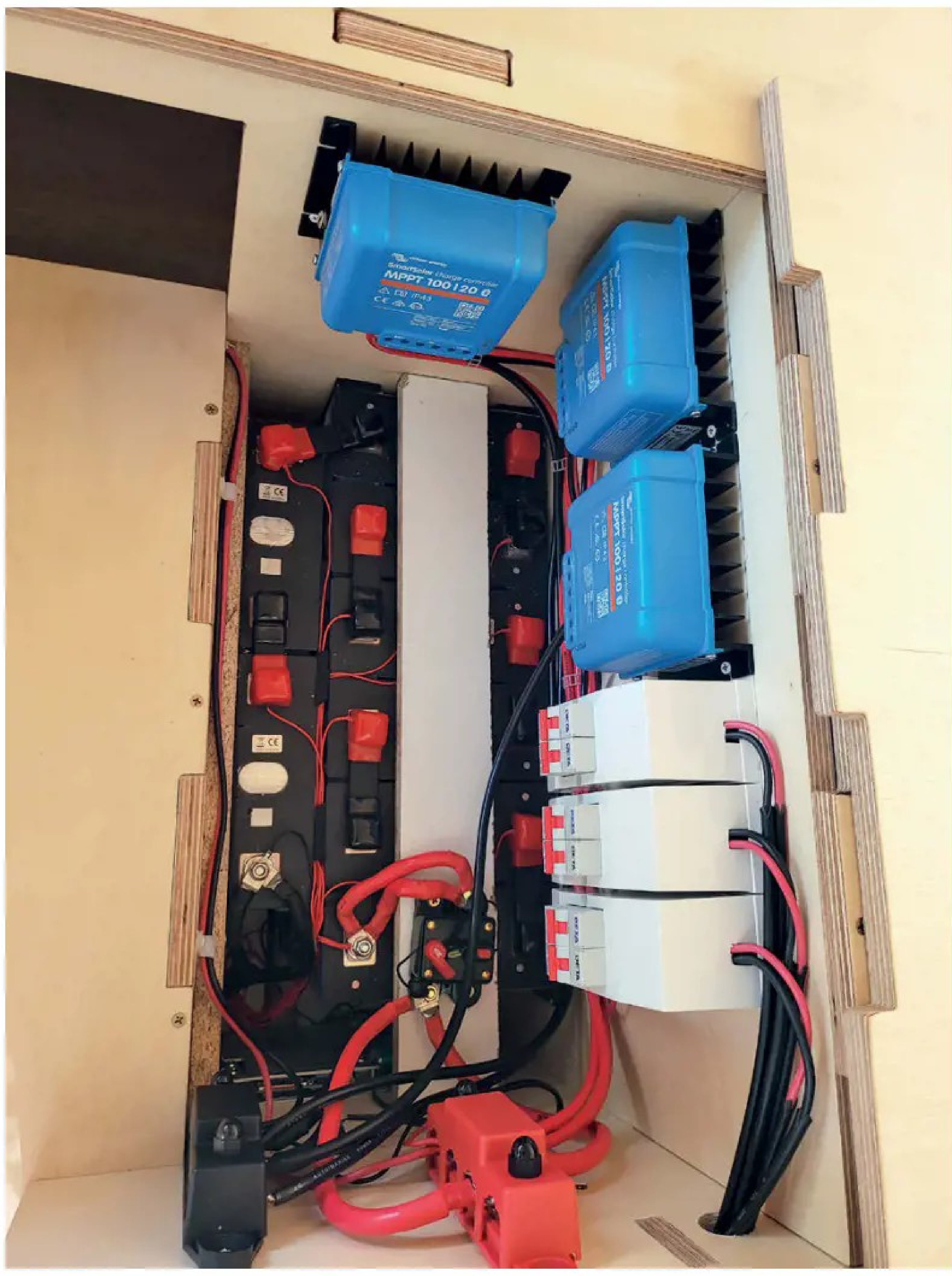
A few tests showed that the charger is about 90% efficient, with 10% lost between the mains and the van battery. The majority of the losses are from the inverter built into the Kangoo.

I conducted a sequence of tests on range and efficiency at different speeds, with the main result being that the battery still had about 30kWh left of the original 33kWh. Not too bad after 85,000km! The best efficiency of 14-15kWh/100km (22.5-24kWh/100mi) was at 50-60km/h (30-35mph). It read about 17.5kWh/100km (28.2kWh/100mi) at 80km/h (50mph) and over 25kWh/100km (40kWh/100mi) at 110km/h (68mph). It is interesting to get used to nearly one-foot driving with the regenerative braking.

I performed another test in hilly terrain, taking the van to the Picadilly Circus pass through the Brindabellas, a voyage about 50km (31 miles) long and 1000m vertical. At the top of the mountain, the consumption had increased to 22kWh/100km (45kWh/100mi), but on returning home, it had dropped back to around 15kWh/100km (24kWh/100mi), having regained most of the energy used to ascend.

This was with careful driving, trying to keep the efficiency indicator out of the red, even if it meant going at only 30km/h (19mph) on the steep parts of the ascent. The regenerative braking certainly is effective.

To sum up the efficiency/range tests, keeping to 50km/h (30mph), I got a range of about 220km (137 miles), but at 80km/hr (50mph), it drops to around 150km (93 miles). These results agree with the USA Electric Vehicle Design Base (EVDC) range estimate for the Kangoo ZE of 160km (100 miles). There is real optimism in Europe with the New European Driving Cycle (NEDC) that claims a range of 270km (168 miles), while the Americans take a more realistic view.



The inside of the timber cabinet, which contains the Victron MPPT solar chargers, circuit breakers, busbars, battery charger etc, as shown in Fig.1 opposite. The eight batteries sit under a piece of wood on which the main circuit breaker is fitted.

Attaching the solar panels

According to the Australian Design Rules (ADR) for loads carried on vehicles using public roads, an overhang of 1200mm without flags is acceptable both front and rear. For side protrusions, 150mm on each side is allowed beyond the vehicle's width.

This meant the Longi 510W panels were a good fit, so we decided on having three lengthwise on the roof, with the first in line with the top of the windscreen and about 500mm of overhang at the rear. MORID Pty Ltd did the design using the roof rack attachment points (three on each side). The main challenge was the roof loading rating of the Kangoo, which is just 100kg.

Having three 25.3kg panels means that the whole roof rack structure could weigh only 24.1kg and had to be strong enough to hold the panels.

A plastic 3D printer was employed to print the fittings for the prototype. We then attached them to the roof to verify their stability, size and appropriateness. Having passed this first hurdle, the design was sent off to PCBWay for machining out of aluminium. These guys are really good and they have never disappointed us. The six adaptors were finished and sent to us. Perfect!

Assembling the panels into an aluminium frame, drilling holes in the roof and attaching them to the van took some time. We just managed to get it a few millimetres under the protrusion rules.

We were then faced with the one task we had been avoiding: drilling holes in the roof to get the cables from the panels into the van so they could be connected to the interior electronics. We took the plunge, drilled the holes in the roof (sorry Mr Renault) and fitted the grommets.

As usual, after the cables were slipped through and the connectors attached, we found that we had forgotten to slip a clip on the connector; oh dear! We had to desolder the connectors, attach the wayward clip, then resolder the connectors.

The Maximum Power Point Trackers (MPPTs)

The solar panels do not charge the Kangoo's battery directly, as the onboard charger does not support charging from low-voltage DC. Instead, the solar panels charge a secondary 24V battery that we installed (more on that later), and that battery runs an inverter that feeds the onboard EV charger – see Fig.1.

Our solar panels put out about 50V and 10A, and we need to charge a 24V battery, so a conversion is necessary, conserving as much power as possible. By chopping the input voltage at around 100kHz, small inductors (or coils) and an electronic circuit called a buck converter can reduce the voltage without wasting too much power. As a result, the output current is higher than the input current.

If we are charging the 24V battery at, say, 27V, the charge current would be 18A minus the losses from the buck converter, which are only around 5% nowadays (ie, 95% conversion efficiency). MPPT is needed to get the most power from the panels, as the voltage/current curve has a peak that moves depending on the ambient conditions. We want to manage the panel voltage to keep it at that point while doing the buck conversion.

The MPPT chargers also continuously monitor the battery SoC to provide the correct charging profile. A few years ago, such circuits comprised lots of individual components and were pretty expensive, but now a single chip can carry out most of the operations and the price of MPPT chargers has fallen dramatically. It pays to shop around!

The secondary battery

Once again, this was a learning experience. For safety, we decided on Lithium Iron Phosphate ($LiFePO_4$) cells since they are less likely to fail than Lithium Manganese Nickel cells (and if they do fail, it's usually less spectacular). However, they have a lower energy density.

The next choice was the voltage. Using a 12V DC battery would require high currents and hence significant Joule losses, so we went for 24V. Should we use a series/parallel arrangement of 12V batteries or build our own 24V system from 3.2V prismatic cells? If the latter, we would need a battery management system (BMS) to balance the voltages of all the cells and prevent overcharging and overdischarging.

I tried both approaches and started by purchasing four 12V 135Ah batteries. These were bought at different times during the COVID-19 years, and we soon discovered that we needed to get a balancing system, so we purchased that as well. It worked, but it was a clunky solution, so off to AliExpress to purchase eight 3.2V 320Ah PWOD prismatic cells and a 24V BMS.

These took some months to arrive, and we eagerly assembled them with the BMS attached to a 20A charger and waited until the BMS cut out. We then discharged it into a bathroom heater via an inverter and surprise, we only got 275Ah. We charged it again and only got 275Ah the second time, so what should we do?

Messaging the PWOD AI was highly frustrating, as it was impossible to have a coherent discussion. They finally offered $26 (£13) off the next purchase, or we could send them back at our expense. Sigh. So we swallowed the bitter pill and realised how the price could be so low – caveat emptor. We would have to make do with 6.7kWh of stored energy, 14% lower than expected.

The inverter

There are a great many DC/AC inverters on the market. The first one I bought was from Victron and it is installed in our solar off-grid shed. Actually, I did buy a few smaller inverters before that for use in the car and for

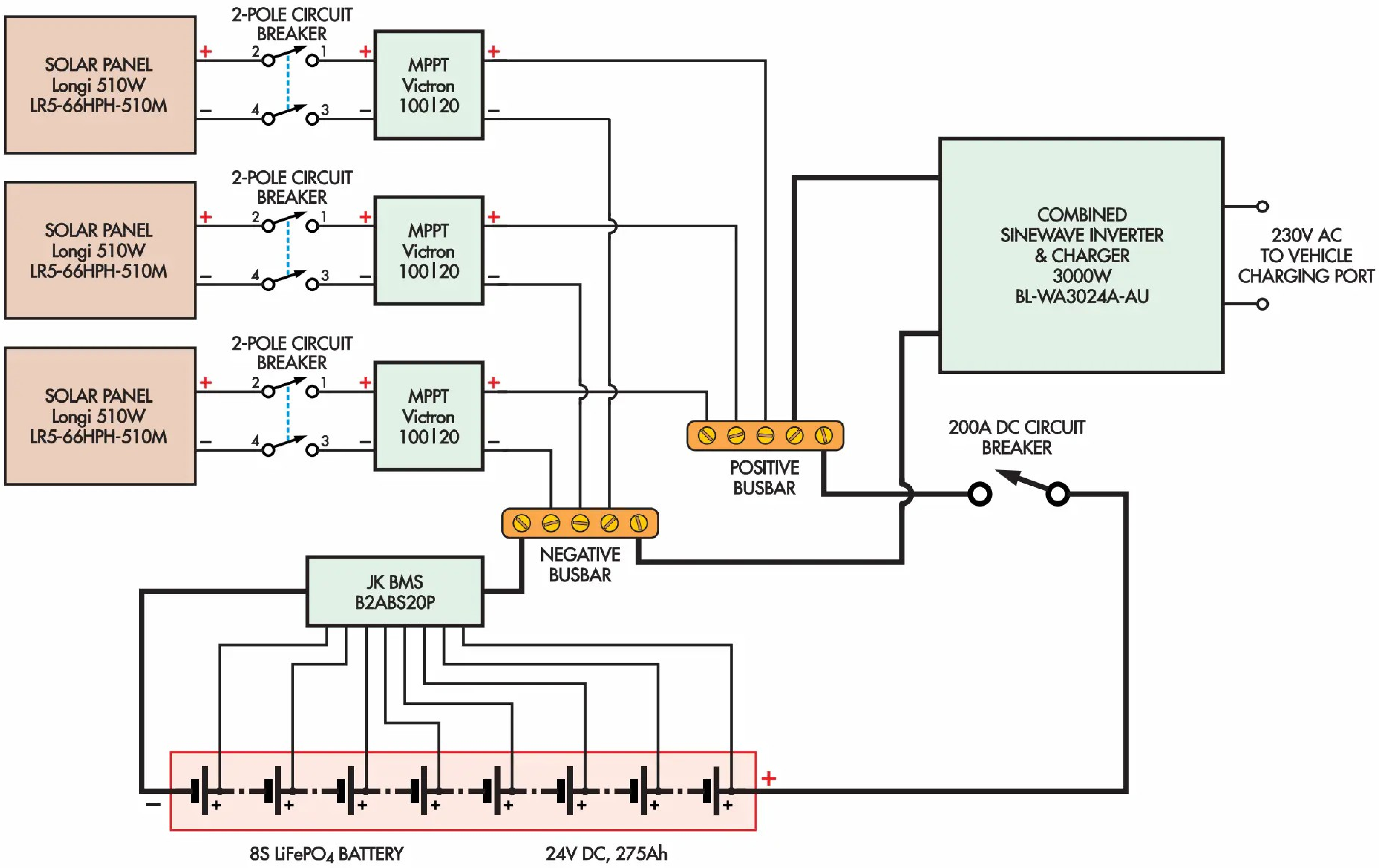

SOLAR PANEL
Longi 510W
LR5-66HPH-510M
SOLAR PANEL
Longi 510W
LR5-66HPH-510M
SOLAR PANEL
Longi 510W
LR5-66HPH-510M
2-POLE CIRCUIT BREAKER
2-POLE CIRCUIT BREAKER
2-POLE CIRCUIT BREAKER
MPPT
Victron
100|20
MPPT
Victron
100|20
MPPT
Victron
100|20
COMBINED
SINEWAVE INVERTER
& CHARGER
3000W
BL-WA3024A-AU
230V AC
TO VEHICLE
CHARGING PORT
POSITIVE BUSBAR
NEGATIVE BUSBAR
200A DC CIRCUIT BREAKER
JK BMS
B2ABS20P
8S $LiFePO_4$ BATTERY
24V DC, 275Ah

Fig.1: each solar panel has its own MPPT battery charger to maximise charging efficiency. The battery management system ensures the cells remain in balance and are not overcharged or overdischarged.

camping, but nothing in the kW range that we were investigating.

Pretty much all the inverters now use chopped voltages rather than heavy transformers, making them quite compact. We needed 3kW continuous and 6kW peak (for a few seconds), with a charger, and we got those capabilities for well under $1000 (£500). However, we soon realised that the inverter's internal charger could only draw a maximum of 2300W from the wall (230V @ 10A).

Since it was to be used for charging the Kangoo, we purchased a second inverter rated at 4kW continuous and 8kW peak for around the same price.

We tested the batteries and inverter before installation to verify that everything was operating as expected. With everything installed in the van and the solar panels connected, the isolating circuit breakers were flipped on and, hooray, the Bluetooth app on our Android phones showed the voltage, current and power being delivered by each of the three panels. The BMS showed the battery charging.

Charging the Car

The last step was to charge the car with solar panels. On the first try, using the 24V battery, inverter and 10A home charger, the car refused to charge. The charger was blinking; after reading the manual, we realised that the error was related the Earth connector on the socket.

Most inverters have Earth wiring, however, most of the time, it is floating. To solve this problem, we connected the Earth and Neutral inside the inverter and the car started charging.

Using only the 24V battery for charging the car, the inverter would draw around 90A and could add 6.5kWh (~40km or 25 miles of range) to the car. Using a fully-charged 24V LFP battery and solar panels on a sunny day at the same time, the solar panels provide around 50A and the battery around 40A, adding up to 16kWh (~100km or 62 miles of range) to the car in one day.

You can't do that two days in a row, though, as the secondary battery would be discharged at the end of the first cycle, and it needs to be charged initially to provide so much energy to the EV battery.

The camper conversion

Our group had some discussions and decided that the Kangoo would be ideal for camping. Since micro-campers are popular, we took the van to Kata Camperbox in Sydney to do their magic. As can be seen from the photos, the fittings are all real timber, and the result is a true beauty to behold.

There is a pull-out kitchen, a slide-out fridge that runs from the 24V battery and a space large enough for two electric bikes that can also be used as a sleeping space. It is about the same size as a business-class bed on an aircraft. To get an idea if everything would work, we took a camping/cycling trip to the Orroral Valley campsite that had recently reopened after the bushfires.

This 55km (34 mile) trip from Canberra was successful; all the subsystems worked, and nothing fell off the van. My electric bike fitted in the van OK. However, I must say that I am not a great fan of sleeping in enclosed spaces, even those at the pointy end of an aircraft. I know; first-world problems!

Planning for a trip

In Australia, the Bureau of Statistics has determined that the average passenger vehicle travels a smidgen over 10,000km (6200 miles) a year, an average of a bit under 30km (18 miles) per day, well within the parameters of our Solely Solar van. The van would have to be left out in the sun all day; still, rooftop parking is generally the last to fill up, so perhaps that is not too much of a drawback.

So what sort of a trip could be made with our little Solely Solar Renault Kangoo ZE, just relying on solar generation of electricity? The Kangoo has 30kWh in its primary battery and 6.7kWh in the secondary battery. Assuming you have solar panels on the roof of your house connected to a home battery, it is simple to just charge the Kangoo at home without paying for grid energy.

You could have a separate solar system for charging the car, but then you could argue that you are losing around 10¢/kWh by not selling any excess power back to the grid power supplier. Then again, nothing is completely free.

However, if you would like to travel further than a few tens of kilometres (eg, to work and back), you need to do a bit of planning, especially if you want to get back in under a week. And there are limitations on how the remote charging is carried out.

As mentioned, the solar panels alone cannot supply the full power necessary for charging via the inverter; they need to be supplemented with power from the solar batteries. Arriving at a campsite in the evening, the solar battery will generally be full, allowing the 6.7kWh (ignoring losses) to be transferred to the traction battery in a few hours while it is dark.

The next morning, the panels will start charging the empty solar battery and will absorb around 4.5kWh by midday, at which time the inverter can be brought into play, allowing both the solar electrons and the secondary battery electrons to flow into the traction battery for the daylight that remains.

Given a sunny summer day, the solar panels would provide around 9kWh, so the Kangoo would need about 3½ days to fully recharge if exhausted. So, with judicious planning and good weather, you could take a long weekend and travel within a radius of around 180km (112 miles) from your house and return, paying virtually nothing for the trip. Not too shabby! PE

The van with solar panels being used for camping for the first time. Removing the bike frees up space to sleep inside.

Practical Electronics

UK readers SAVE £1 on every issue

SUBSCRIBE NOW!

01202 087631

Take out a one-year subscription and save more than £10 over the year.
Even better – save over £1 per issue if you subscribe for two years – a total saving of £26.

Overseas rates represent exceptional value

You also:
- **Avoid any cover price increase for the duration of your subscription**
- **Get your magazine delivered to your door each month**
- **Guarantee a copy, even if your local newsagent sells out**

Order by:
- **Phone or post with a cheque, postal order or credit card**
- **Or order online at: www.electronpublishing.com**

Practical Electronics

SUBSCRIPTION PRICES

Subscriptions for delivery direct to any address in:

UK: 6-months £33.99, 12-months £59.99, 24-months £114.99

Europe Airmail: 6-months £38.99, 12-months £69.99, 24-months £129.99

Rest Of The World Airmail: 6-months £46.99, 12-months £84.99, 24-months £164.99

Cheques (in **£ sterling only**) payable to ***Select Publisher Services Ltd*** and sent to: Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Tel: 01202 087631
Email: pesubs@selectps.com

Also via our online shop at**: www.electronpublishing.com**

Subscriptions start with the next available issue. We accept MasterCard or Visa.
(For past issues see the Back Issues page.)

DIGITAL SUBSCRIPTIONS

Electronic subscriptions are available from £29.99 for 12 months, more details at: **www.electronpublishing.com**

SUBSCRIPTION ORDER FORM

☐ **6 Months: UK £33.99, Europe £38.99 (Airmail), Rest of the World £46.99 (Airmail)**

☐ **1 Year: UK £59.99, Europe £69.99 (Airmail), Rest of the World £84.99 (Airmail)**

☐ **2 Years: UK £114.99, Europe £129.99 (Airmail), Rest of the World £164.99 (Airmail)**

To: **Practical Electronics Subscriptions**
PO Box 6337
Bournemouth BH1 9EH
United Kingdom

Tel: 01202 087631
Email: pesubs@selectps.com

I enclose payment of £ ....
(cheque/PO in £ sterling only)
payable to ***Select Publisher Services Ltd***

☐ Please charge my Visa/Mastercard

My card number is: ....
Please print clearly, and check that you have the number correct

Card Security Code .... Valid From Date.......................
(The last 3 digits on or just under the signature strip)

Card Ex. Date ....

Name ....

Address ....

Post code .... Tel. ....

Email ....

Subscriptions start with the next available issue.

Precision Electronics

Part 5: Noise

So far, this series has mostly been concerned with errors arising from component matching and unwanted currents and fixed voltages. There is another type of unwanted signal that can cause all sorts of problems that we refer to as noise. So, how can we quantify its effects on our circuits and reduce the resulting errors?

By Andrew Levido

In electronics, the term “noise” can refer to any form of unwanted signal that masks a signal of interest. This includes noise that is imposed upon a circuit from external sources such as radio-frequency interference or mains hum, which can be reduced or eliminated by filtering, shielding or other design techniques.

However, there are sources of noise that are intrinsic to the components themselves. They come from within the circuit, not outside, so they cannot be reduced by shielding.

In this article, we will look at this type of truly random noise that is caused by various physical phenomena within the components we use. As we have discussed in earlier articles, precision electronics design is all about understanding and quantifying the sources of uncertainty, and noise is another error source that we cannot always ignore.

To understand noise, we will have to start with a bit of theory, but I will try to keep it to a minimum. We will then get into the practical side with a full noise analysis of a simple audio amplifier. Let’s begin by looking at the main types of noise of concern to electronics designers.

Johnson noise

Johnson noise, sometimes also called Nyquist or thermal noise, is essentially the electrical signal produced in lossy components by the random movement of charge carriers (usually electrons) due to temperature. Remember that temperature relates to the motion of atoms and other elements of a material; they are only still when the material is at absolute zero.

For example, a 10kΩ resistor at room temperature will develop a noise voltage across its terminals of around 1.3μV RMS if measured with a high-impedance AC voltmeter with a 10kHz bandwidth. If we were to short-circuit the resistor, we would measure a noise current of about 130pA (1.3μV ÷ 10kΩ) over the same bandwidth.

This phenomenon was first described by John Bertrand Johnson in 1927 and characterised by Harry Nyquist in 1928. Nyquist showed that the noise voltage density in a resistor is given by the equation $V_{RMS} = \sqrt{4kTRf_b}$, where k is Boltzmann’s constant (1.38×10^{-23}), T is the absolute temperature in Kelvin, R is the resistance and f_b is the bandwidth in hertz (Hz).

This highlights the first important thing to keep in mind when discussing noise: we can only quantify a noise voltage or current if we also specify a bandwidth over which to measure it. If we don’t know the bandwidth of concern, we can only describe noise in terms of a voltage or current per unit frequency, called the voltage or current noise density.

The units of voltage noise density are $V/\sqrt{Hz}$ and those for current noise density are $A/\sqrt{Hz}$.

You have to be careful to distinguish between an absolute value of noise (an RMS [root-mean-squared] voltage or current) and its density. The relationship between them is analogous to the relationship between the mass and density of a material. Density is a property intrinsic to the material, but the mass of an object made of the material depends on the amount of it we are dealing with in a specific case.

The noise voltage density of a resistor, for example, is a property of the resistor at a given temperature, but the noise voltage developed across it depends on the bandwidth we use to measure it (or over which it has an effect). In the case of Johnson noise (and any white noise, as we will see below), the relationship between noise density and voltage is the square root of the bandwidth.

Johnson noise is an inescapable result of the thermal agitation of electrons that occurs anywhere that charges are free to move. Fortunately, you can normally ignore the Johnson noise generated in conductors like wires, since their resistance is so low that any noise that they may contribute is negligible.

In fact, mostly lossless devices like capacitors and inductors (up to a point) do not contribute to the overall noise in a circuit. They can, however, impact bandwidth and therefore can influence the noise voltage or current.

Shot noise

Johnson noise occurs even when no current is flowing. On the other hand, shot noise occurs because a flowing current is made up of discrete ‘chunks’ of charge (electrons or holes). If the moving charges act independently of each other, the resulting randomness of the current flow causes noise.

This phenomenon does not occur in metallic conductors, where the moving electrons influence each other and therefore don’t act randomly. It does occur in semiconductors, though; for example, when charge carriers are diffusing across a semiconductor junction.

The shot noise density is given by the formula $i_n = \sqrt{2qI_{DC}}$ and is expressed in units of $A/\sqrt{Hz}$. Here, q is the charge on an electron (1.6×10^{-19}C) and I_{DC} is the average current. The RMS value of shot noise is therefore $i_{RMS} = \sqrt{2qI_{DC}f_b}$.

This means that a steady 1A current passing through a semiconductor junction will see a random variation of 57nA RMS when measured over a

10kHz bandwidth. That corresponds to about 0.057ppm (parts per million) – small enough to be immaterial in most situations.

Shot noise gets relatively larger as the current reduces because there are fewer moving charge carriers. For example, a 1µA current will have a superimposed shot noise of 57pA RMS, which corresponds with 57ppm; it is becoming more significant.

$^1/_f$ noise

Shot and Johnson noise are both types of 'white noise', ie, they carry equal power per unit of frequency (hertz) across the spectrum. This is why we can simply multiply the noise density by the square root of the bandwidth to calculate the noise voltage or current.

$^1/_f$ noise (sometimes called flicker noise) differs from Johnson or shot noise, which are the result of atomic-level physical phenomena. $^1/_f$ noise is created by a variety of mechanisms (not all well understood) relating to materials and construction techniques.

$^1/_f$ noise has the defining characteristic of having an equal energy per decade of spectrum. In other words, it has the same power in the 1-10Hz range as it does in the 10-100Hz and 1-10kHz ranges. This means the power per hertz is inversely proportional to frequency – hence the $^1/_f$ name.

Noise with this power spectrum is known as 'pink noise' and it occurs in a wide variety of places, including the flow of traffic, ocean currents and the loudness profile of classical music!

From an electronics perspective, $^1/_f$ noise does occur in some resistors depending on their construction (carbon composition resistors were notoriously bad for this) but it can usually be safely ignored in modern resistors. It can become significant in op amp circuits, which is why we need to know about it.

Burst and avalanche noise

There are two other sources of noise in electronics that are worth mentioning: burst noise and avalanche noise. Burst noise, also known as popcorn noise, is a low frequency (<100Hz) 'popping' phenomenon caused by manufacturing imperfections in semiconductor materials. It used to be a problem in the early days of integrated circuits, but improved manufacturing processes have all but eliminated it.

Avalanche noise is similar to shot noise but occurs during the reverse breakdown of semiconductor junctions. Its amplitude can be very high, so it is often used when we want to deliberately to create an analog white noise source. We don't usually operate our circuits with semiconductor junctions in reverse breakdown, so we can ignore avalanche noise most of the time.

There is one major exception: zener diodes with voltage ratings above about 5.5V operate in a controlled avalanche breakdown mode. Those rated below 5.5V use the Zener effect, which is a different phenomenon altogether (interestingly, a 5.6V zener diode uses a mixture of both!). If you have these in your circuits, you may have to take avalanche noise into account.

Now we have covered the sources of noise, we have to consider one more important point that will allow us to analyse noise in practical circuits.

Gaussian distributions

Johnson, shot and $^1/_f$ noise are all considered Gaussian, which means the amplitude of instantaneous voltage is distributed according to a Gaussian, sometimes called 'normal', curve as shown on the left of Fig.1. The average amplitude of noise is zero, but the peak value at any given instant will be a matter of probability.

Higher-amplitude excursions are less likely than those close to the mean due to the 'bell' shape of the Gaussian distribution. The probability that the instantaneous voltage will be between any two values is given by the area under the curve between those values.

Statisticians love these curves, but I don't find them a particularly intuitive way to look at amplitude. I prefer the relative occurrence chart on the right-hand side of Fig.1. The vertical scale is the fraction of time the instantaneous voltage will exceed some multiple of the RMS voltage.

For example, the instantaneous amplitude will exceed twice the RMS voltage approximately 4.6% of the time, but will exceed four times the RMS voltage only 0.006% of the time.

You can also see from the occurrence chart that the instantaneous noise voltage will be below the RMS value approximately ⅔ of the time. This, plus the fact that the different sources of noise in a given circuit will be uncorrelated (they behave completely

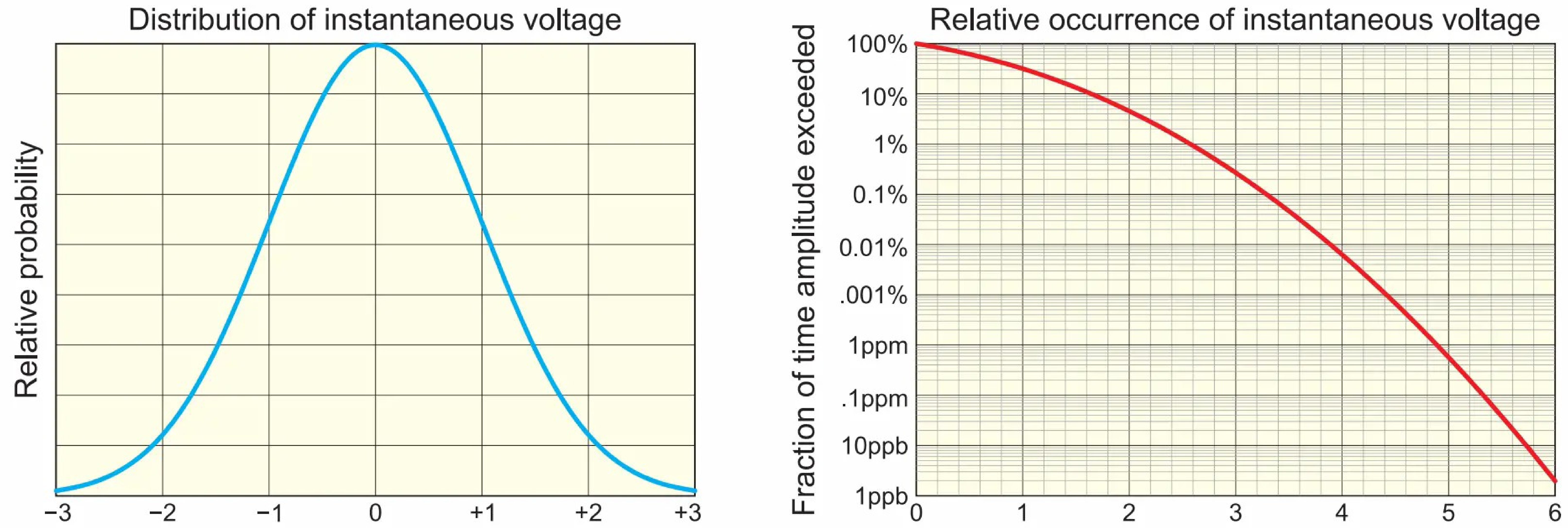


Fig.1: the probability of a given instantaneous voltage occurring in white noise is distributed according to a Gaussian curve (left). The relative occurrence curve on the right instead shows the fraction of time that the peak voltage will exceed a specific amplitude, which I find easier to understand.

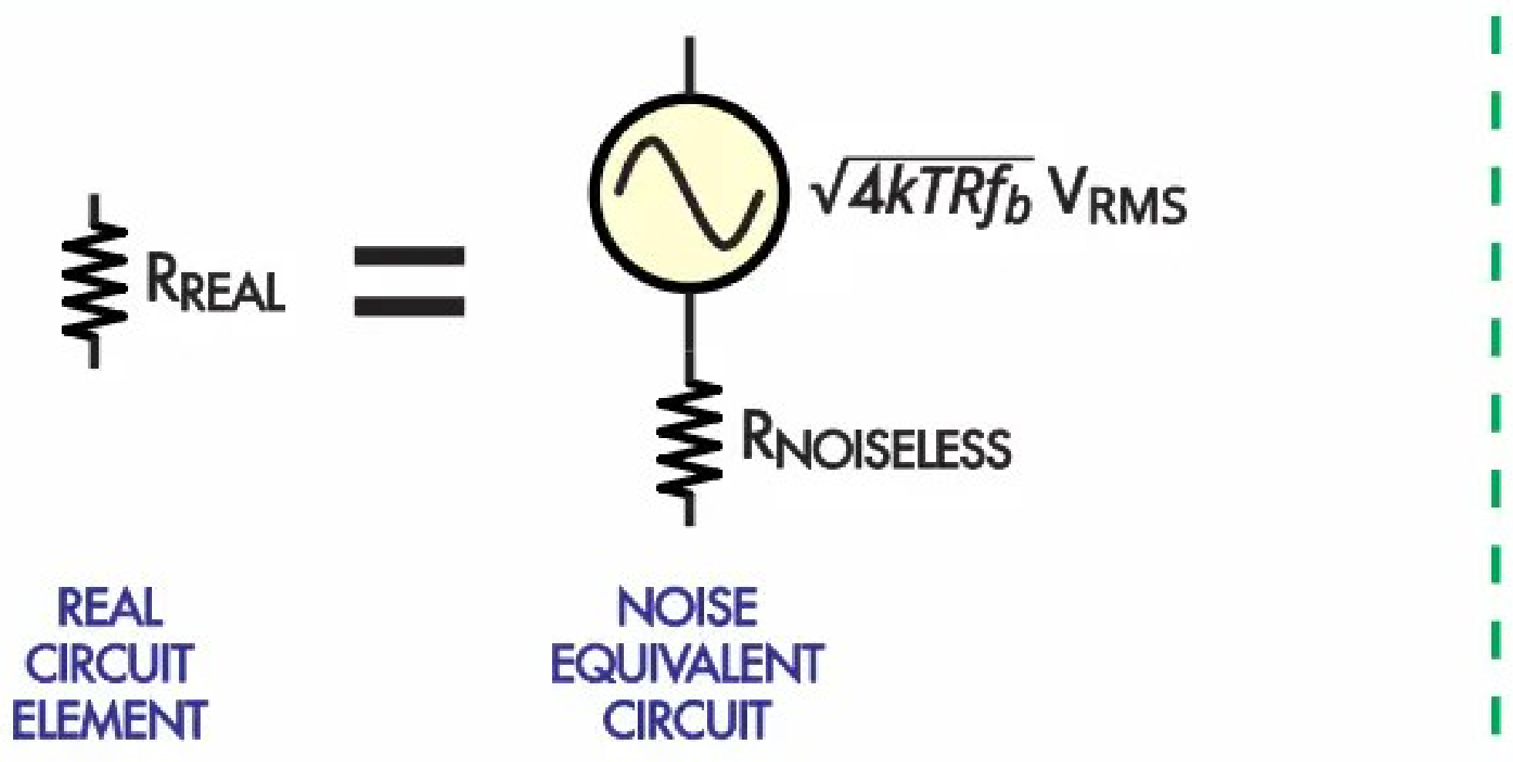


Fig.2: the noise equivalent circuit of a resistor is a noiseless resistor in series with a voltage source with a value given by the Johnson noise equation.

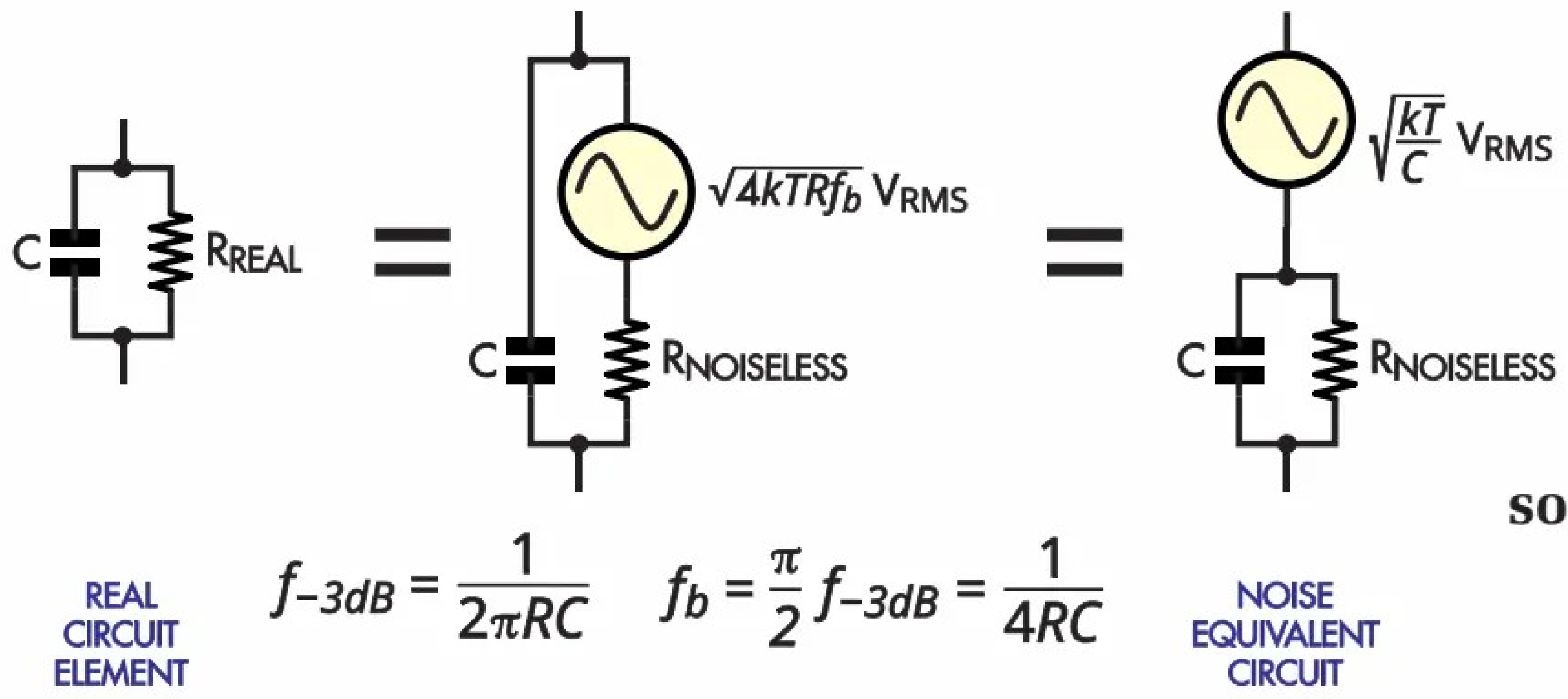

C
R_{REAL}
=
$\sqrt{4kTRf_b}$ V_{RMS}
C
$R_{NOISELESS}$
=
$\sqrt{\frac{kT}{C}}$ V_{RMS}
C
$R_{NOISELESS}$
REAL CIRCUIT ELEMENT
$f_{-3dB} = \frac{1}{2\pi RC}$
$f_b = \frac{\pi}{2} f_{-3dB} = \frac{1}{4RC}$
NOISE EQUIVALENT CIRCUIT

Fig.3: the noise equivalent circuit for a parallel RC circuit. The resistor and capacitor form a low-pass filter for the resistor's Johnson noise source. The resulting noise voltage depends only on the temperature and the capacitor value.

independently), means that adding the RMS values of different noise sources will over-estimate the resulting noise.

So, instead of adding noise in the normal arithmetic way, we add noise as the root sum of squares. This means we square the values, add them together, then take the square root to arrive at a value that is statistically equivalent to the sum. If we are adding more than two noise sources, we just extend the sum by adding more squared values before taking the square root.

The root sum of squares method leads to a shortcut that can help simplify noise calculations significantly. If one of the quantities being added is smaller than another by a factor of 10 or more, you can ignore it altogether with very little resulting error. For example, a 10μV and 1μV source sum to 10.05μV using root-sum-of-squares, so we could simply ignore the 1μV source in that case.

A simple example

Noise calculations in circuits can get quite complex since almost every component contributes to or shapes noise. For this reason, it is important to take a step-by-step approach, breaking the circuit down into manageable chunks. For each 'chunk' of circuit, we need to analyse the 'noise equivalent circuit' to calculate the total noise.

We must understand the noise equivalent circuits of the components to accomplish this.

The noise equivalent circuit of a resistor (Fig.2) is a good place to start. We have already seen that a resistor will exhibit Johnson noise with a voltage of $\sqrt{4kTRf_b}$ so it can be modelled by a noiseless resistor in series with a voltage source of that value. Resistors in parallel or series can be reduced to a single equivalent resistor, then converted to the noise equivalent circuit.

We mentioned above the capacitors don't themselves contribute noise, but Fig.3 shows the noise equivalent circuit of a capacitor in parallel with a resistor. If we replace the resistor with its noise equivalent circuit from Fig.2, we have a noise source feeding a simple single-pole RC low-pass filter. This filter will reduce the noise bandwidth and thus the noise voltage seen at the terminals of the RC pair.

The -3dB bandwidth of the RC filter will be 1 ÷ (2πRC), but we can't just use this in noise calculations since such a filter lets through frequencies higher than the corner frequency, albeit in an attenuated form. We therefore need to convert the -3dB bandwidth to an 'equivalent noise bandwidth' (ENBW), which is the bandwidth of a perfect 'brick wall' filter that lets through the same amount of noise.

The scale factor for a single-pole filter turns out to be π ÷ 2, so the ENBW for a single-pole RC filter is 1 ÷ (4RC).

Substituting this into the equation for Johnson noise gives an expression for the resulting noise voltage: $V_{RMS} = \sqrt{kT \div C}$. A parallel RC circuit therefore has the noise equivalent circuit shown on the right in Fig.3. Notice that the noise voltage is dependent only on the temperature and the capacitor value – the resistor value is irrelevant! Noise can be weird sometimes.

Op amp noise equivalent circuit

A typical op amp exhibits Johnson and shot noise with a flat power spectrum, as well as $1/f$ noise, which has a power spectrum biased towards lower frequencies as shown in Fig.4. At low frequencies, the $1/f$ noise dominates, while at high frequencies, the Johnson and shot noise dominate.

At some frequency, f_c, the amplitudes of these two noise components will cross over. When specifying op amp noise, manufacturers condense everything down to three things: a figure or graph for f_c, an input noise voltage density (e_n) referred to the non-inverting input, and an input noise current density (i_n) at each input.

Fig.5 shows the noise equivalent circuit of an op amp. It consists of a noiseless op amp with noise sources at its inputs. These current sources

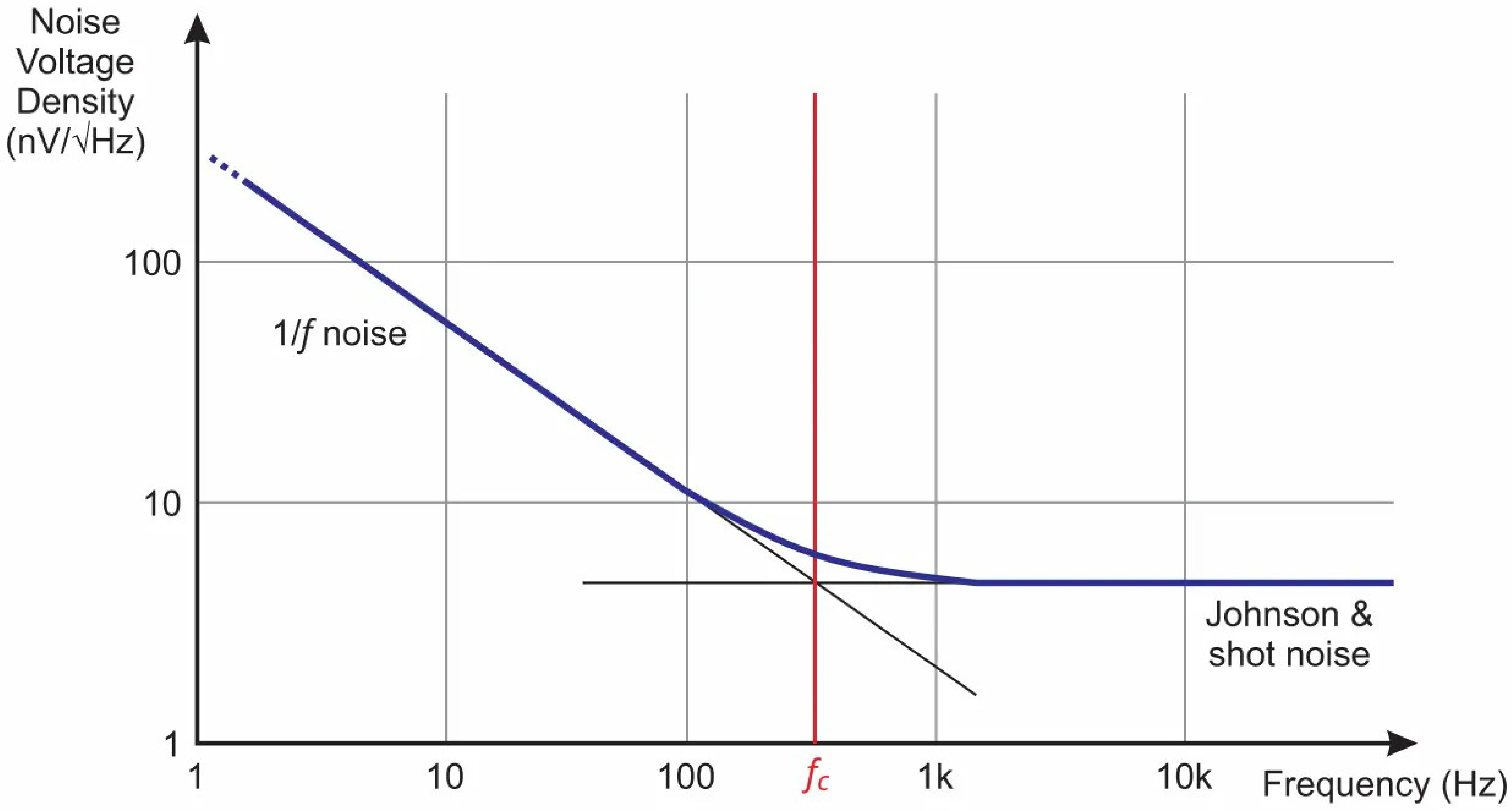


Fig.4: op amps exhibit both pink ($1/f$) noise, which dominates at low frequencies, and white (Johnson and shot) noise, which dominates at high frequencies. The frequency at which they cross over is the noise corner frequency, f_c. This figure is usually specified in the data sheet.

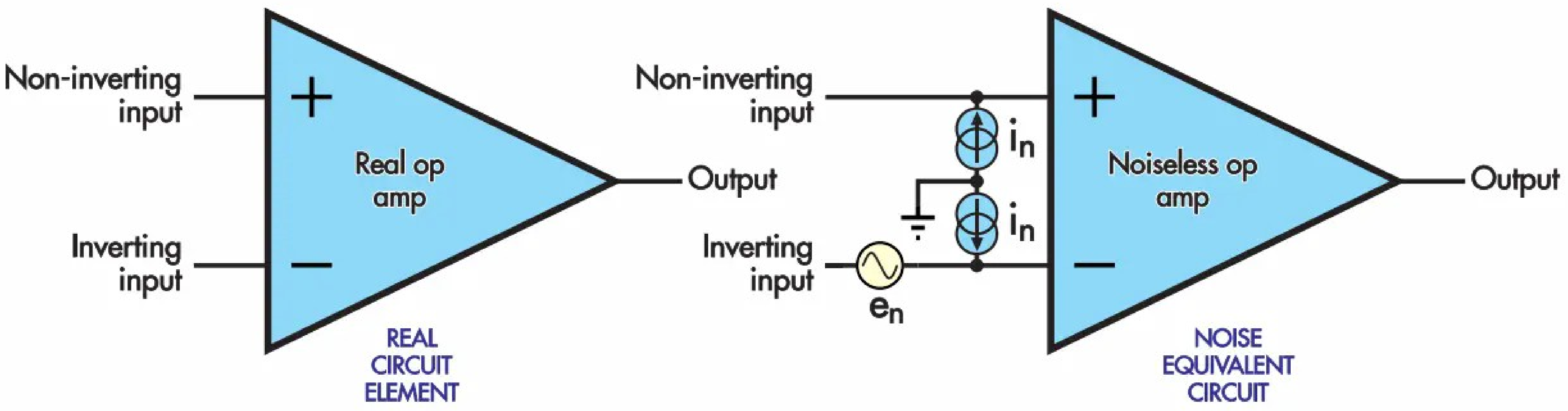


Fig.5: the noise equivalent circuit of an op amp includes noise current sources associated with each input of a noiseless op amp, plus a noise voltage source in series with its non-inverting input.

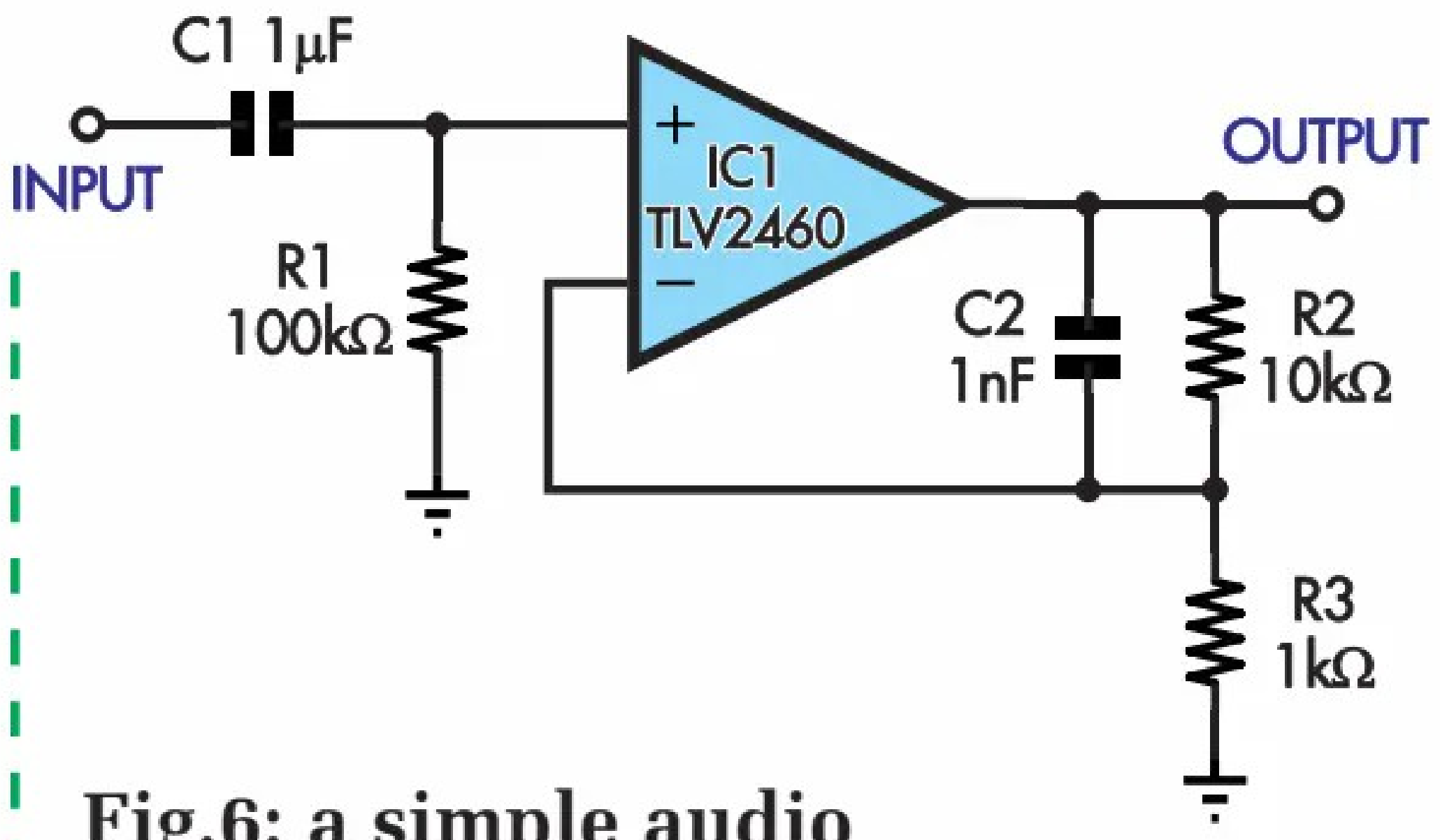


Fig.6: a simple audio amplifier stage circuit we will analyse for noise performance.

produce noise voltages across the source resistances at the op amp's inputs. The noise crossover frequency is used together with these sources to calculate the overall op amp noise.

A practical example

Let's apply this to a real-world example. Consider a simple op-amp based audio amplifier circuit, as per Fig.6. The input signal is AC-coupled to the op amp via C1. This forms a high-pass filter with R1, which has a -3dB cutoff frequency of about 1.6Hz. The op amp is configured as a non-inverting amplifier with a signal gain of 11.

C2 rolls off the amplifier's frequency response at around 16kHz. The example circuit uses a TLV2460 general-purpose rail-to-rail input/output (RRIO) op amp with $e_n = 11\text{nV}/\sqrt{\text{Hz}}$, $i_n = 0.13\text{pA}/\sqrt{\text{Hz}}$ and $f_c = 150\text{Hz}$.

We will analyse the noise in this amplifier in a step-by-step manner, using the principle of superposition. This analysis technique allows us to analyse linear circuits with multiple sources by calculating the effect of each one in isolation and adding the results.

We replace any voltage sources we are ignoring with short circuits, and any current sources we are ignoring with open circuits. We will build up a noise budget table (Table 1) as we go.

Before we start, we need to define the nominal bandwidth over which we will calculate the noise. Since our circuit has a single-pole 1.6Hz high-pass filter at the input and a single-pole 16kHz low-pass filter around the op amp, our overall bandwidth is well defined. However, just like the parallel RC case above, we have to calculate the equivalent noise bandwidth, since the roll-offs are far from abrupt.

Recalling that the ENBW scale factor for a single pole filter is $\pi \div 2$, the lower ENBW limit becomes 1Hz and the upper one becomes 25kHz. So, the overall noise bandwidth of the circuit is 25kHz – 1Hz ≈ 25kHz.

We start our analysis at the non-inverting input of the op amp. Since the source voltage is short-circuited, R1 and C1 are in parallel as far as noise is concerned.

Fig.7(a) shows the noise equivalent circuit of this part of the circuit. There are three noise sources: the R1/C1 parallel noise voltage, the op amp's voltage noise source and the current noise source, both at the non-inverting input of the op amp. We will look at each in turn and use the superposition principle.

Fig.7(b) shows the R1/C1 source. We have already seen that the noise voltage in this case is $\sqrt{kT \div C}$. Plugging in a temperature of 300K (26.85°C) and the 1µF value of C1 gives a noise voltage of 64.3nV. This calculation is shown in row 1 of the noise budget table (Table 1).

Next, we have the op amp's noise voltage, shown in Fig.7(c), which is given by the voltage noise density ($11\text{nV}/\sqrt{\text{Hz}}$) and the bandwidth. We have seen that for white noise, we simply multiply the noise density by the square root of the circuit bandwidth to get the voltage. However, we know that the op amp produces pink $^1/_f$ noise up to 150Hz – well within our 1Hz to 25kHz bandwidth.

We take this into account by applying a modification factor to the bandwidth, a bit like we did to determine the ENBW from the -3dB bandwidth.

When the bandwidth of interest straddles f_c, we have to use the formula $f_b + f_c\log_e(f_h \div f_l)$ to calculate an equivalent bandwidth. In this equation, f_b is the nominal bandwidth, f_c is the noise corner frequency, f_h is the upper bandwidth limit and f_l is the lower limit of bandwidth.

So the op amp noise bandwidth evaluates to 26.5kHz instead of 25kHz. Plugging this into the op amp's input noise density gives us a noise voltage at 1.79µV.

We can now consider the noise voltage generated by the op amp's current noise. This is a bit trickier. Fig.7(d) shows the superposition circuit for this source. We have a current source in parallel with a resistor (and a capacitor,

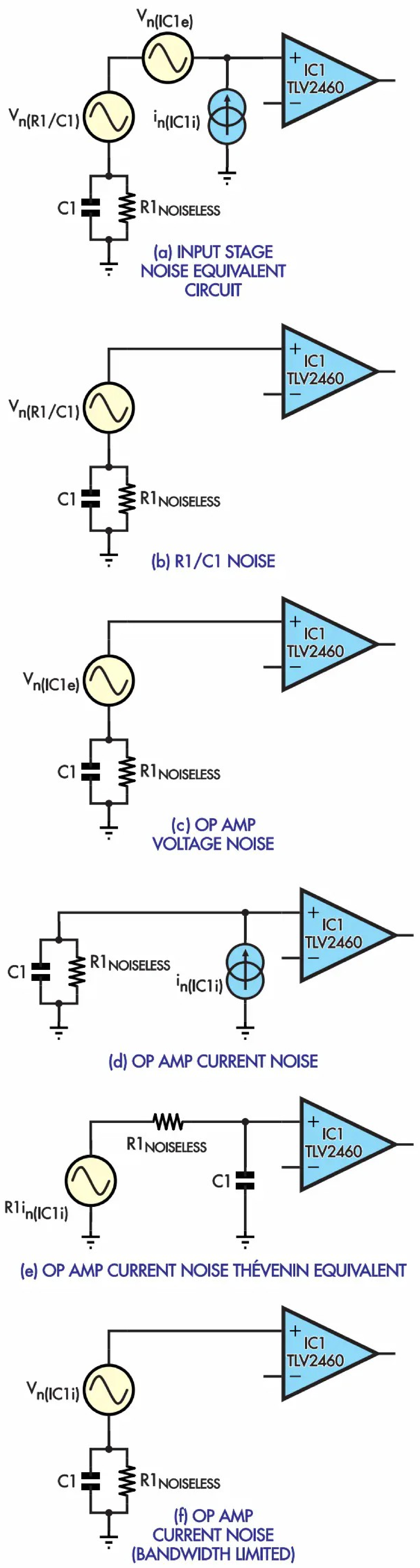
NEW!

5-year collection: 2019-2023

All 60 issues from Jan 2019 to Dec 2023 for just £44.95
PDF files ready for immediate download

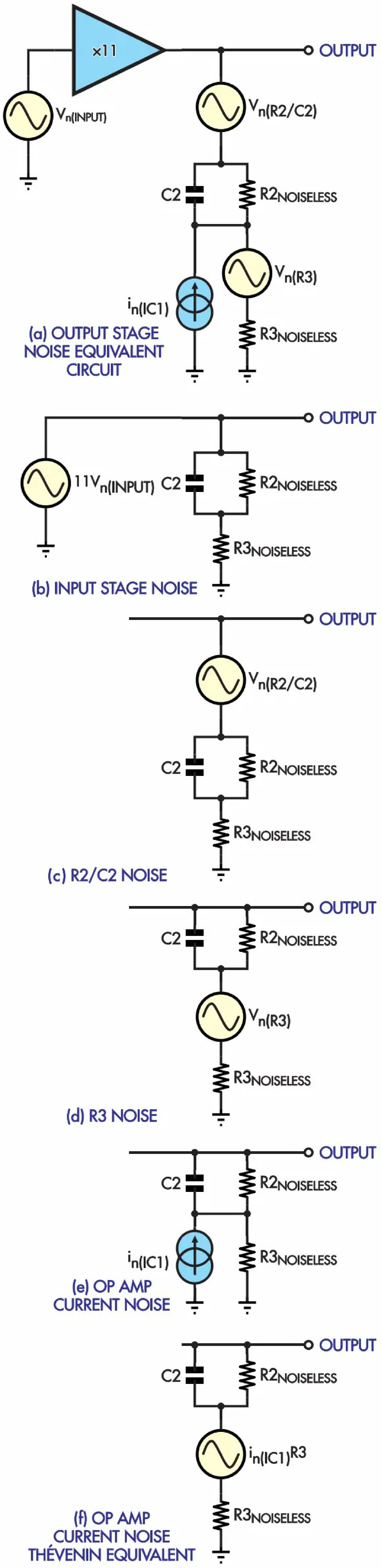
See page 67 for further details and other great back-issue offers.

Purchase and download at: www.electronpublishing.com



V_n(IC1e)
IC1
TLV2460
V_n(R1/C1)
i_n(IC1i)
C1
R1_NOISELESS
(a) INPUT STAGE NOISE EQUIVALENT CIRCUIT
IC1
TLV2460
V_n(R1/C1)
C1
R1_NOISELESS
(b) R1/C1 NOISE
IC1
TLV2460
V_n(IC1e)
C1
R1_NOISELESS
(c) OP AMP VOLTAGE NOISE
IC1
TLV2460
C1
R1_NOISELESS
i_n(IC1i)
(d) OP AMP CURRENT NOISE
R1_NOISELESS
IC1
TLV2460
C1
R1i_n(IC1i)
(e) OP AMP CURRENT NOISE THÉVENIN EQUIVALENT
IC1
TLV2460
V_n(IC1i)
C1
R1_NOISELESS
(f) OP AMP CURRENT NOISE (BANDWIDTH LIMITED)

Fig.7: the process of analysing the input stage noise equivalent circuit for Fig.6(a) shows all the noise sources together and the subsequent circuits show how the noise contribution of the individual sources are evaluated.



×11
OUTPUT
V_n(INPUT)
V_n(R2/C2)
C2
R2_NOISELESS
V_n(R3)
i_n(IC1)
R3_NOISELESS
(a) OUTPUT STAGE NOISE EQUIVALENT CIRCUIT
OUTPUT
11V_n(INPUT)
C2
R2_NOISELESS
R3_NOISELESS
(b) INPUT STAGE NOISE
OUTPUT
V_n(R2/C2)
C2
R2_NOISELESS
R3_NOISELESS
(c) R2/C2 NOISE
OUTPUT
C2
R2_NOISELESS
V_n(R3)
R3_NOISELESS
(d) R3 NOISE
OUTPUT
C2
R2_NOISELESS
i_n(IC1)
R3_NOISELESS
(e) OP AMP CURRENT NOISE
OUTPUT
C2
R2_NOISELESS
i_n(IC1)R3
R3_NOISELESS
(f) OP AMP CURRENT NOISE THÉVENIN EQUIVALENT

Fig.8: the process for analysing the noise sources at the op amp's output is similar to that for the input. This time, there are four noise sources, including the input noise multiplied by the op amp's noise gain.

but let's park that for a moment), so we can replace these with the Thévenin equivalent voltage source in series with the resistor, as shown in Fig.7(e).

In case you are not aware of Thévenin's theorem, it states, "Any linear electrical network containing only voltage sources, current sources and resistances can be replaced by a voltage source in series with a resistance." It is a powerful tool for simplifying circuits and well worth reading about if you don't understand it (see https://w.wiki/9XaJ).

The current noise density is $0.13\text{pA}/\sqrt{\text{Hz}}$ and the resistance is 100kΩ, so the Thévenin noise voltage density is $13\text{nV}/\sqrt{\text{Hz}}$.

Now let's bring the capacitor back into the picture. This forms a low-pass filter with R1, which will impact the bandwidth we should use to calculate the RMS voltage at the op amp's input. The filter's ENBW is given by 1 ÷ (4RC), as we saw above, so we use this figure (2.5Hz) in the calculation. The resulting voltage will be 20.6nV (line 3 of the Table 1).

To complete the superposition process, we just have to add all three voltages together using root-sum-of-squares to get a single equivalent noise voltage at the non-inverting input of the op amp.

Since the op amp's input noise voltage is more than an order of magnitude higher than either of the other two, we can safely ignore the others. This leaves us with a total voltage noise voltage at the op amp input of 1.79µV RMS.

Now we turn to the op amp output. Fig.8(a) shows the equivalent circuit. This time, there are four sources of noise voltage: the output noise of the op amp due to the amplified input noise we just calculated, two resistor noise voltages (R2 and R3) and the current noise at the op amp's inverting input. We use superposition again to calculate each contributing part of the noise voltage individually.

Fig.8(b) shows that the first of these is easy; it is just the 1.79µV input noise voltage multiplied by a gain of 11, giving 19.7µV. For a non-inverting amplifier, the noise gain is the same as the signal gain, since the noise source is on the same input as the signal. This isn't necessarily true for all op amp configurations, so you will need to scrutinise each circuit for noise sources.

For example, for a standard inverting amplifier configuration, the noise gain equals the signal gain plus one. Remember that the op amp's voltage noise equivalent is always referred to the non-inverting input.

The noise due to R2/C2, as shown in Fig.8(c), is also easy to calculate since this is just another RC parallel circuit that we are already familiar with. The noise voltage here is 2.03µV.

The noise due to resistor R3, shown in Fig.8(d), is also easy using the formula for Johnson noise on page 60. This works out to be about 643nV.

The noise voltage due to the op amp's current noise can be calculated using the Thévenin equivalent circuit shown in Figs.8(e) & 8(f). In this case, the noise voltage works out to be 20.6nV. The only trick here is to use the 26.5kHz augmented bandwidth to account for the $1/f$ noise.

The total noise at the output of the circuit is calculated by root-sum-of-squares of the voltages. It turns out to be 19.8µV RMS. Since the circuit was designed to deliver signals at 1V RMS, the signal-to-noise ratio (SNR) is $20\log_{10}(1\text{V} \div 19.8\mu\text{V}) \approx 94\text{dB}$.

Power supply noise

This analysis so far ignores power supply noise. Power supply noise will be coupled into the op amp's output to some degree, although op amp designers try hard to maximise the rejection of such noise. The op amp's power supply rejection ratio (PSRR) defines the degree to which a disturbance on the power supply rail reaches the output.

The TLV2460 has a power supply rejection ratio (at 25°C) of >80dB up to 20kHz. This means any power supply noise will be attenuated by a factor of 10,000 over the frequencies we care about.

To calculate the power supply noise contribution, you reduce the RMS noise present on the power supply by the PSRR and add the result as you would for any other noise voltage source on the op amp's output.

Put another way, if we can keep the power supply noise in our example circuit under 20mV RMS, the op amp's PSRR will reduce this to 2µV at the op amp output. This is one tenth of the ~20µV of noise we just calculated, so will make no meaningful contribution to the overall circuit noise.

Minimising noise

We have seen that noise is an inescapable phenomenon, with causes linked to the fundamentals of physics. There is nothing we can do to eliminate it, short of cooling our circuit down to absolute zero. However, you can minimise the noise in any circuit using a few techniques.

They may not all be possible or relevant in your application, but the following ideas are worth considering.

1. Reduce bandwidth: we have seen that noise voltage depends highly on bandwidth and that reducing bandwidth reduces noise amplitude. You may not be able to reduce the overall bandwidth of the circuit, but even limiting bandwidth in parts of the circuit may help.

2. Use oversampling: this is a kind of bandwidth reduction. If you are measuring a noisy quantity with an ADC, you can take multiple samples and average the results. Since noise is Gaussian with zero mean, the average of several samples tends toward zero as the number of samples increases.

3. Minimise gain: the gain elements in your circuit amplify input-side noise voltages. Use the minimum gain necessary to amplify your signal.

4. Use lower value resistors where possible. A 1kΩ resistor has a noise voltage density of ~4nV/√Hz, compared to ~40nV/√Hz for a 100kΩ resistor. A 100Ω resistor is even better at ~1.3nV/√Hz.

5. Choose the right op amps. There is a huge range of 'low noise' op amps and their headline specifications don't always tell the full story. Low voltage noise density does not always mean low $1/f$ noise and vice versa. Depending on your bandwidth, you might need to balance one parameter with another.

6. Pay attention to power supplies. Power supply noise can be coupled into your signal path in all sorts of ways. Linear regulators are generally quieter than switch-mode ones (or you could use a switch-mode regulator with a linear post-regulator).

7. Pay attention to power supply and ground routing, especially where high current circuits are present on the same board. Use decoupling capacitors thoughtfully and consider using LC filters or capacitance multipliers to create a low noise supply to particularly sensitive portions of the circuit. Keep high-current ground networks separate from signal grounds.

As a concrete example, refer to our circuit shown in Fig.6. We calculated its SNR as being close to 94dB. If we intended to use it to process an audio signal, we probably want to reduce the noise a bit, for an SNR of at least 100dB.

That could most effectively be achieved by using a lower noise op amp. An op amp like the NE5534, for example, with a voltage noise density of 3.5nV√Hz, a current noise density of 1.5pA√Hz and an f_c of about 1kHz might be a better choice.

A quick estimate using these figures gives an op amp voltage noise (line 2 of the table) of 0.66µV compared to 1.79µV, and the overall circuit noise figure reduces to about 7.5µV, giving an SNR of just over 102dB.

Table 1 – a noise budget for our example audio amplifier circuit

Line	Noise Source	Figure	Notes	Result (RMS)
1	R1/C1 (100kΩ, 1µF)	7(b)	Parallel RC: $V_N = \sqrt{kT \div C}$	64.3nV
2	Op amp voltage noise (11nV/√Hz, 150Hz)	7(c)	Bandwidth straddles f_c so use $V_N = e_N\sqrt{f_b + f_c \log_e(f_h \div f_l)}$	1.79µV
3	Op amp current noise (0.13pA/√Hz)	7(d)(e)(f)	Thévenin equivalent and LPF: $e_N = i_n R1$, $f_b = 1 \div (4RC)$	20.6nV
4	**Total input noise (Lines 1-3)**	**-**	**Root sum of squares, Line 2 dominates**	**1.79µV**
5	Output noise due to input noise	8(b)	Line 4 times noise gain of 11	19.7µV
6	R2/C2 (10kΩ, 1nF)	8(c)	Parallel RC: $V_N = \sqrt{kT \div C}$	2.03µV
7	R3 (1kΩ)	8(d)	Resistor: $V_N = \sqrt{4kTRf_b}$	643nV
8	Op amp current noise (0.13pA/√Hz)	8(e)(f)	Thévenin equivalent: $V_N = i_n R3\sqrt{f_b + f_c \log_e(f_h \div f_l)}$	21.2nV
9	**Total output noise (Lines 5-8)**	**-**	**Root sum of squares, Line 5 & 6 dominate**	**19.8µV**

AUDIO OUT

By Jake Rothman

A switchable Class-A/Class-(A)B amplifier

Sometimes it's worth making a simple experimental discrete circuit for fun rather than a paid job.

I have an old Hacker radio, like the one shown in Photo 1, with an excellent loudspeaker, case and FM section where the audio amplifier PCB was a write off. It was an unusual design running on two PP9 batteries feeding an 8×5 inch elliptical 25Ω Goodmans speaker (shown in Photo 2).

A pair of batteries costs over £10, which was more than a power supply, but I still wanted to retain battery operation for my Radio 4 fix in the bath where they last about three years. For this application, keeping the current draw to a minimum is important, so Class-B operation is required (where there is basically no standing current through the output transistors at idle).

When a mains supply is being used, Class-A will draw more power but will provide better sound quality as it eliminates crossover distortion, where one output transistor stops sinking or sourcing current and the other takes over, which happens at every zero crossing.

That is inevitable in Class-B due to the lack of standing current but is totally absent in Class-A, where both output transistors are conducting all of the time, regardless of the instantaneous output voltage.

Ideally, in a radio that can run from batteries or a mains supply, the amplifier can switch from Class-B (strictly speaking, Class-AB, where there is a small quiescent current) to Class-A automatically when connected to the mains. There's no cheap chip I can find that does this, and Douglas Self's Trimodal amplifier would be overkill.

That made designing a simple circuit worthwhile. This little amp is good practice to prepare for more expensive designs later, the parts cost being only about that of a cup of coffee, so a blow up is of little consequence. It's also useful for other things, such as musical instruments.

Other old radios that this circuit could be used with, having 25Ω speakers and 18V supply rails, include the Robert's 404, Ultra Transistor Six and the Perdio PR25/36/27. The latter was also available as a Lasky's/Henry's Radio kit for *Radio Constructor* magazine.

The circuit

The circuit is shown in Fig.1. It could easily have been made around the time of the original radio in 1973; it uses a four-transistor topology that was popular at the time. This is about the minimum possible to make an amplifier that sounds okay. It gives about 1.3W RMS, sufficient for the high-sensitivity loudspeaker used.

Since there is a very limited open-loop gain of around 200, the distortion in Class-B is high, but in Class-A it sounds much smoother. That's because the effectiveness of negative feedback in cancelling distortion is related to the open-loop feedback divided by the closed-loop feedback, so a higher open-loop feedback (well over 1000 times) is desirable.

Our closed-loop gain (ie, the amplifier's voltage gain) is 25 times, so with an open-loop gain of 200, that only gives

▲ Photo 1: many old radios can benefit from this design. A Hacker radio is shown here next to the green telephone at RetroTech 2024 (photo by Harvey Rothman).

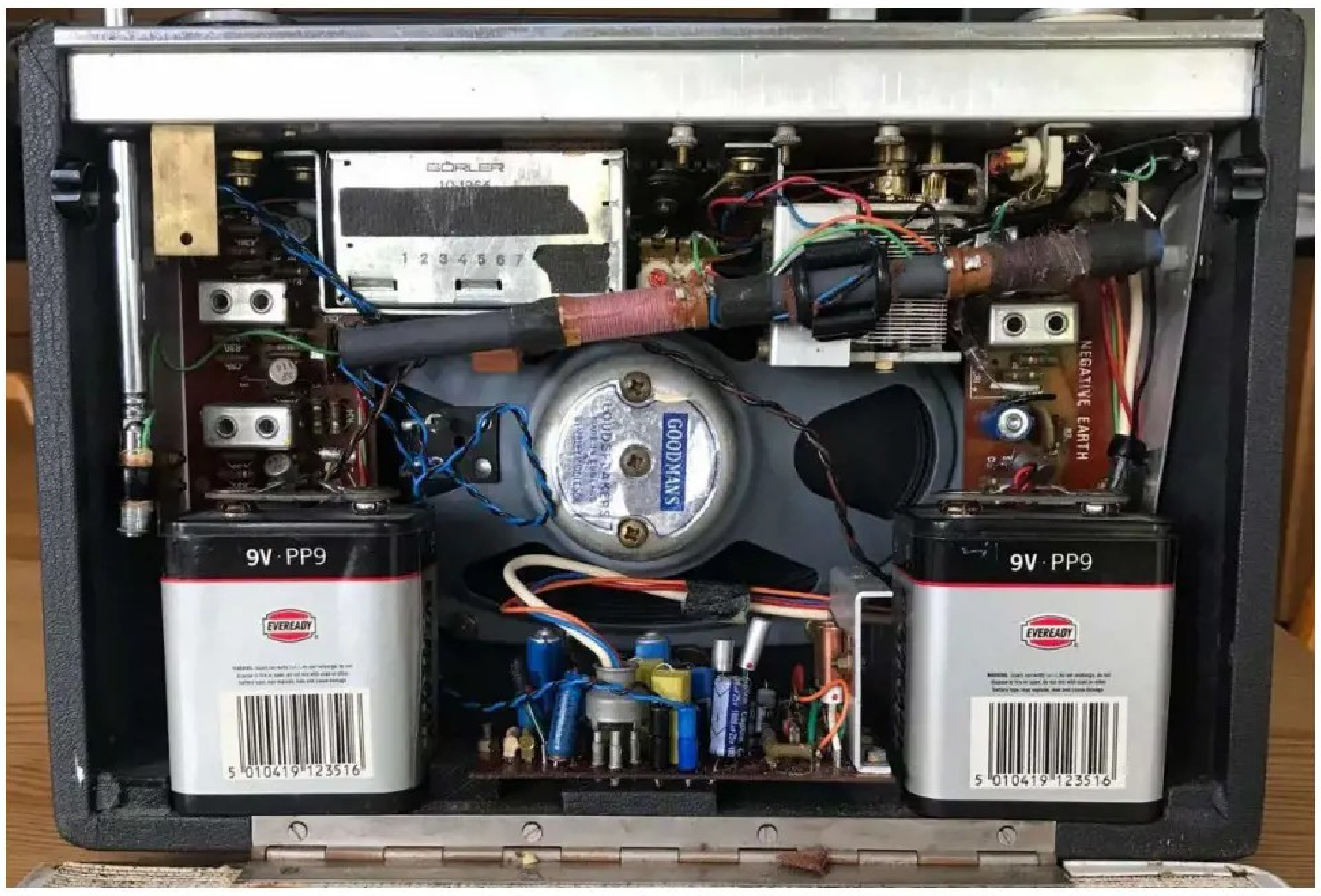


Photo 2: a Hacker radio; the original audio amplifier PCB is at the base of the cabinet between the two large PP9 batteries.

a feedback ratio of 200 ÷ 25 = 8. That means that any distortion generated by the signal path (the non-linearity of the transistors etc) will be reduced by about 87.5% (1 – ⅛). By comparison, if the open loop gain were 2500, the distortion would be reduced by 99% (1 – 1⁄100).

The high-impedance load helps; it would be very difficult to make this circuit sound acceptable with a 4Ω speaker.

TR1 is a common-emitter voltage amplifying stage with a collector current of 4.5mA feeding a push-pull emitter-follower output consisting of TR3 and TR4. The output stage is biased into Class-AB (ie, slight conduction) for battery operation by Vbe multiplier stage TR2, which can be thought of as an adjustable zener diode.

The output stage standing current is set to 3mA by adjusting VR1, giving a total supply current of 7.5mA, which gives good battery life. TR2/VR1 set the quiescent current by setting the voltage between the bases of TR3 & TR4 to a fixed bias of about 1.2V, sufficient to get them just into conduction.

Mode changing

For Class-A operation, around 90mA of collector current for TR3 & TR4 is needed, and this is set by potential divider R5 and R6 feeding the base of TR2. To switch from Class-B to Class-A operation, VR1 has to be switched out. This could be done with an ordinary switch, but in this case I've used a P-channel JFET so that the class switching can be voltage controlled.

JFETs are depletion-mode devices that conduct a little current until they are biased off (similar to valves), so in this case, a positive voltage on the gate will switch it off for Class-A operation. The gate is normally pulled to ground by R4 unless switched into Class-A.

Normally, 90mA of standing current would result in thermal runaway of the output stage. This occurs because the Vbe necessary to bias a transistor on drops with increasing temperature. As the temperature increases, if the bias voltage remains constant, the collector current increases, as does power dissipation, increasing the temperature. This is positive feedback!

This is prevented by a Nelson Pass designed feedback circuit derived from his A40 amplifier design (in *Audio Amateur* magazine, 1978) consisting of R9 and D2. This senses the current across the emitter resistor (R11) and switches on TR2 harder as the output stage current increases.

This, in turn, reduces the bias voltage across the output transistors, reducing the current and ensuring thermal stability. The effect of this circuit is shown in Fig.2.

One problem with these types of output current feedback circuits is that the audio signal can modulate the current, causing second harmonic distortion. This is reduced by using a large bypass electrolytic capacitor (C5) across TR2. This is not very effective at low frequencies, but the distortion of the loudspeaker is going to be much worse below its resonance of around 120Hz.

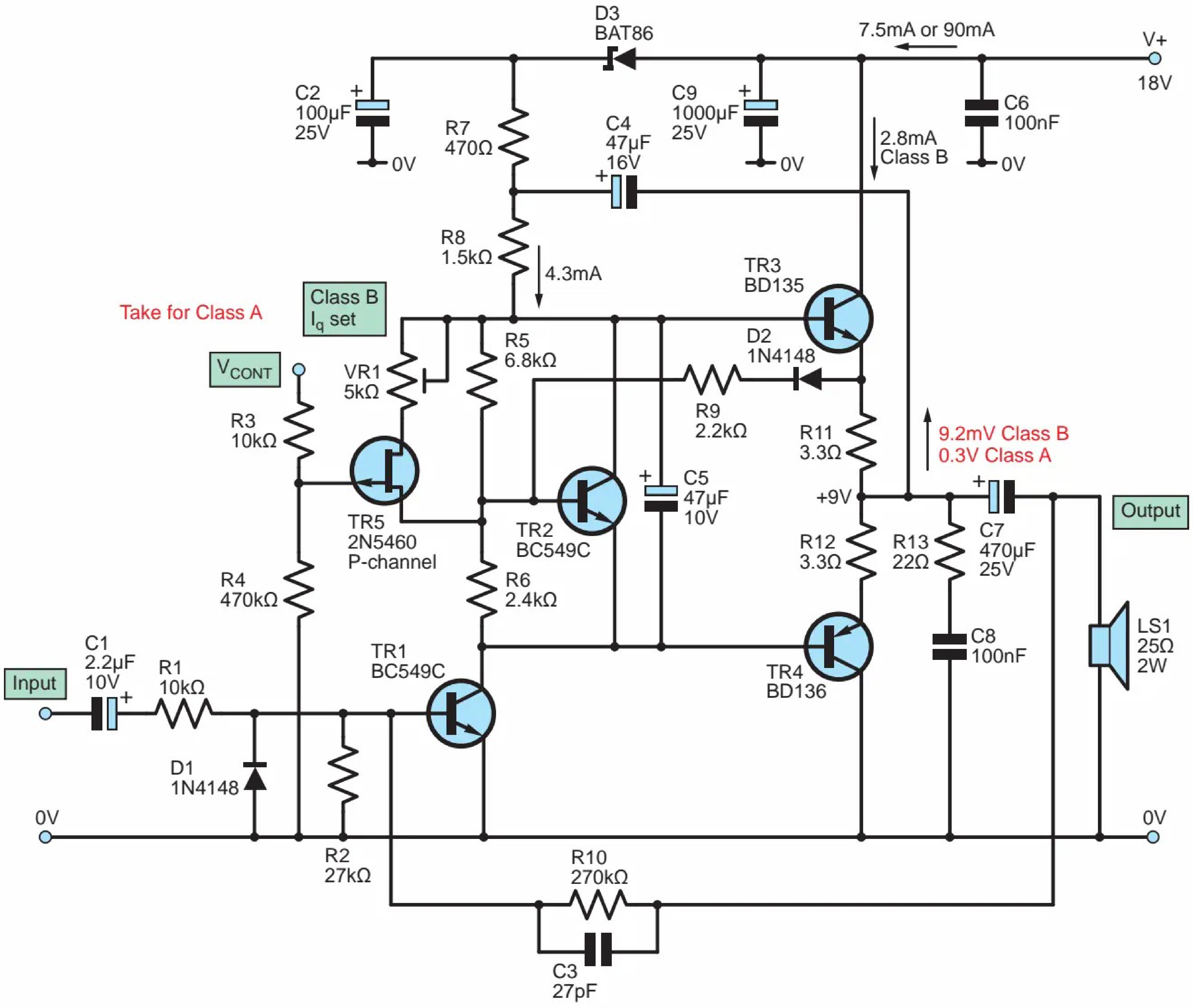


Fig.1: the switchable amplifier circuit. The transistors are all 1970s European types.

D1 exists to ensure a symmetrical path for negative and positive currents through the input circuit (along with TR1's base-emitter diode junction). Otherwise, DC could build up on C1 under overload conditions, such as if it were used for a guitar amp.

Resistors R7 and R8 form a collector load for TR1. This is bootstrapped by C4 to boost the voltage drive and increase the open-loop gain. R8 could be replaced by a 4.5mA current regulator diode or another constant current source for lower crossover distortion, but the maximum power output would be reduced by about 30%.

Diode D3, along with capacitor C2, provides decoupling for the input stage. Normally, a resistor would be used in place of D3, but by using a low forward voltage diode here, such as a schottky type, more output voltage is obtained. Decoupling is very important to minimise distortion when running on battery since the pulsating output current would otherwise modulate the supply voltage.

This is further minimised by placing a large electrolytic capacitor (C9) across the main supply rails.

The closed-loop voltage gain is set by R1 and R10 to around 25× in a similar manner to an inverting op amp circuit, with R10 being the negative feedback resistor. R2 is necessary to bleed off some bias current to set the average output level at half-rail. This may need some adjustment depending on the Hfe and Vbe of transistor TR1.

Lower distortion can be obtained by increasing the value of R1 but this reduces the gain, and with many radios, a preamp gain stage will be required to get a decent output volume. There are a few extra capacitors needed for high frequency stability, such as for phase lead compensation C3, Zobel network C8/R13 and high-frequency decoupling capacitor C6 in parallel with the bulk bypass capacitor, C9.

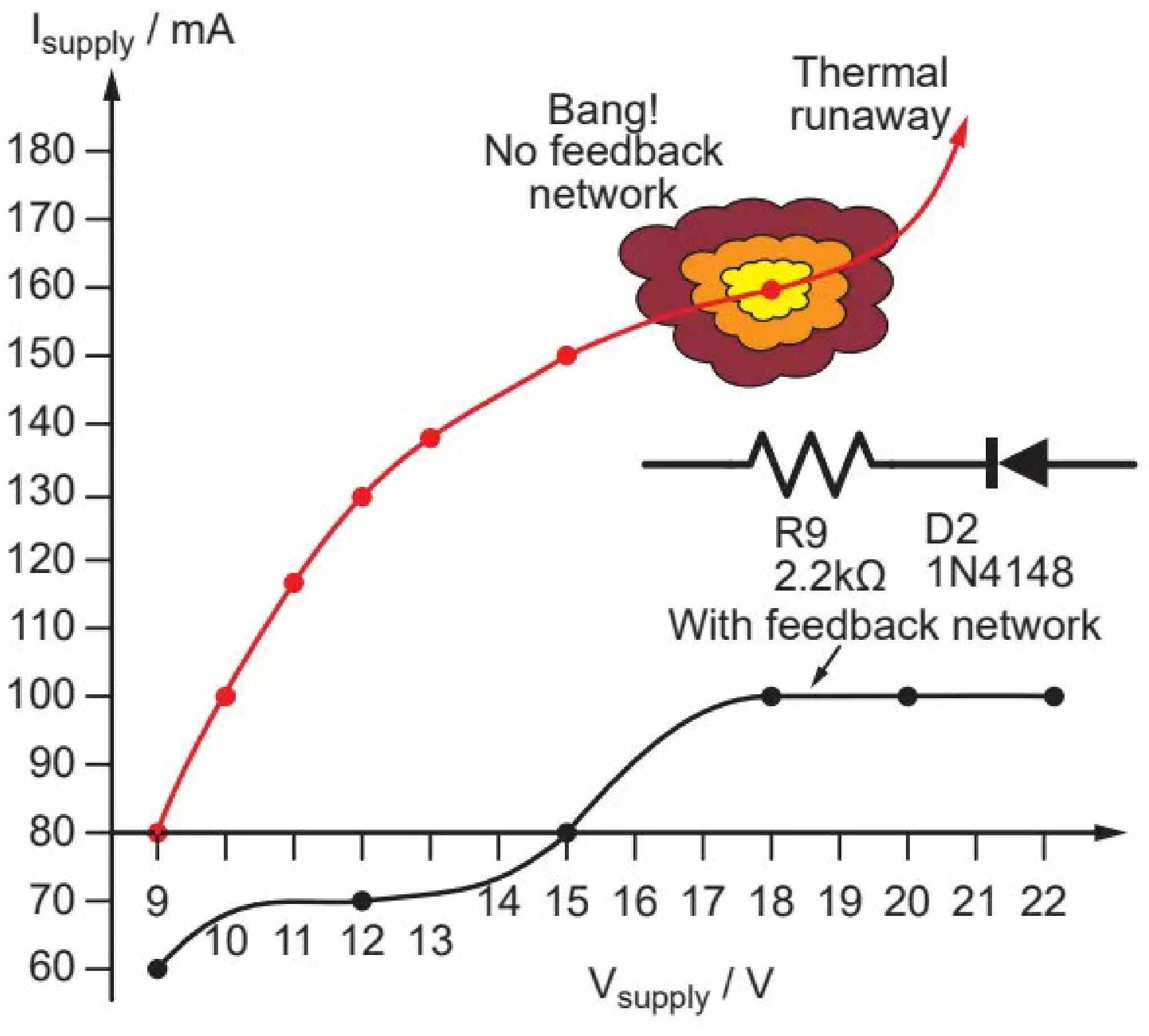


Fig.2: the current stabilising effect of the Nelson Pass quiescent current sensing circuit as the supply voltage varies.

Powering it

Many old radios using germanium transistors have a positive Earth, meaning that all the polarised components in Fig.1 have to be reversed. The PNP transistor becomes NPN and all the others PNP. The JFET has to be an N-channel type. The +18V power rail becomes -18V.

The circuit in Fig.3 could be used to connect the circuit to a battery and power supply with automatic switchover. I used diodes, but for lower voltage drop, Mosfets could be employed instead. However, there would then be the possibility of static charge damage requiring protection (eg, a zener diode between each gate and source).

Construction

As this amplifier is a one-off, for the moment, I just built it on perfboard, as shown in Photo 3. I have yet to design a PCB or rescale the component values for different voltages and output impedances.

In Class-A mode, the output transistors dissipate almost 1W each continuously, so small heatsinks are required (I used flag types). It would also be a good idea to sandwich the bias transistor, TR2, between the output devices for proper thermal feedback. The only unusual part is the high-impedance loudspeaker, which I can supply (jrothman1962@gmail.com). **PE**

Parts List – Switchable Class-A/B Amp

1 4.7kΩ trimpot (VR1)

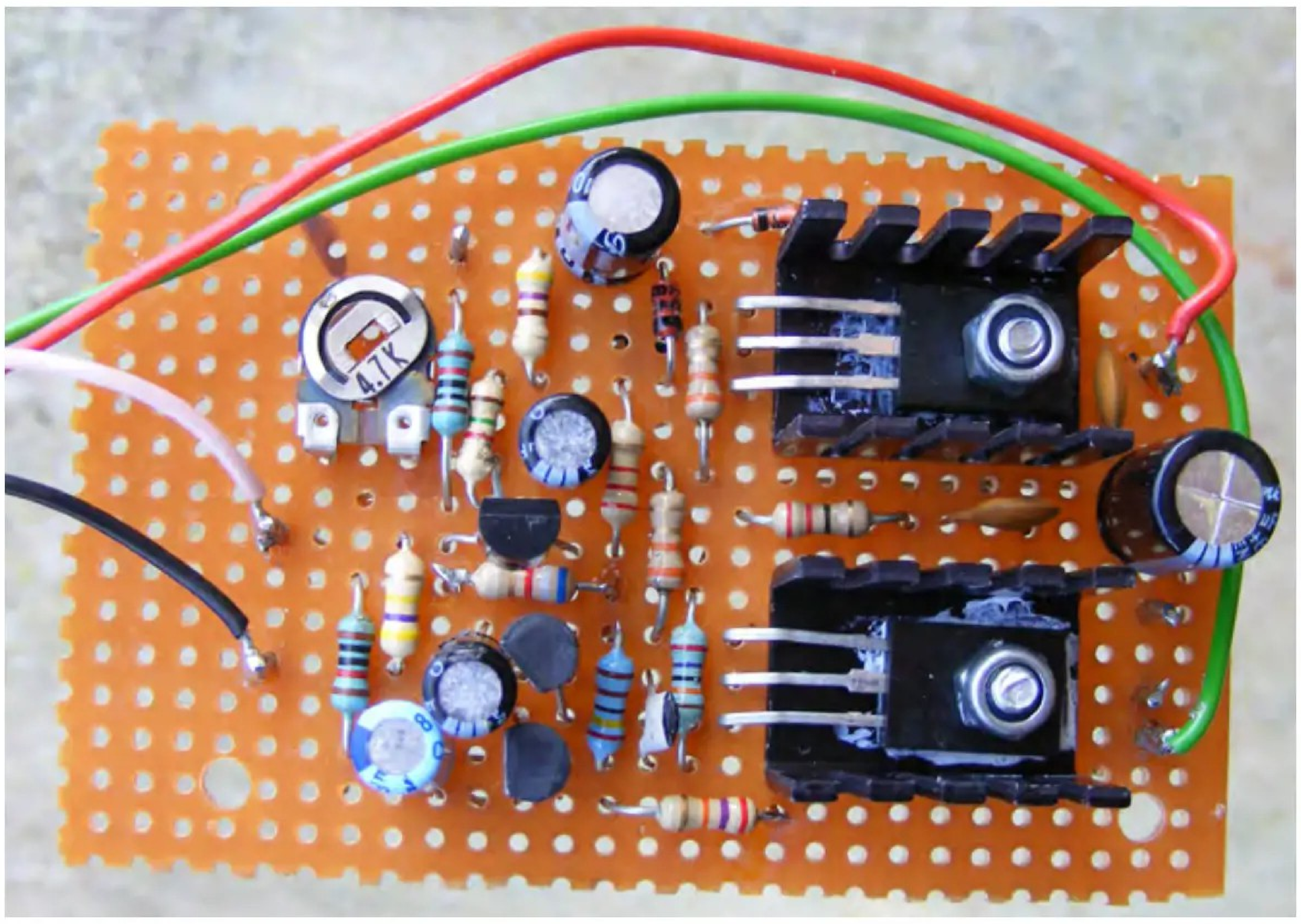
Semiconductors
2 BC549C 30V 100mA NPN transistors (TR1, TR2)
1 BD135 45V 1.5A NPN transistor (TR3)
1 BD136 45V 1.5A PNP transistor (TR4)
1 2N5460 40V 10mA or similar P-channel JFET (TR5)
2 1N4148 75V 200mA signal diodes (D1, D2)
1 BAT86 50V 200mA or similar schottky diode (D3)

Capacitors
1 1000µF 25V electrolytic (C9)
1 470µF 25V electrolytic (C7)
1 100µF 25V electrolytic (C2)
1 47µF 16V electrolytic (C4)
1 47µF 6.3V electrolytic (C5)
1 2.2µF 10V electrolytic (C1)
2 100nF 50V X7R ceramic (C6, C8)
1 27pF ±5% 50V C0G/NP0 ceramic (C3)

Resistors *(all ±5% ¼W carbon film unless noted)*
1 470kΩ (R4)
1 270kΩ ±1% metal film (R10)
1 27kΩ (R2)
2 10kΩ (R1, R3)
1 6.8kΩ (R5)
1 2.4kΩ (R6)
1 2.2kΩ (R9)
1 1.5kΩ (R8)
1 470Ω (R7)
1 22Ω (R13)
2 3.3Ω (R11, R12)

Qty	Value	4-band code	5-band code
1	470kΩ		
1	270kΩ		
1	27kΩ		
2	10kΩ		
1	6.8kΩ		
1	2.4kΩ		
1	2.2kΩ		
1	1.5kΩ		
1	470Ω		
1	22Ω		
2	3.3Ω		

Photo 3: a perfboard prototype of the amplifier. Note the output transistor heatsinks. ▶



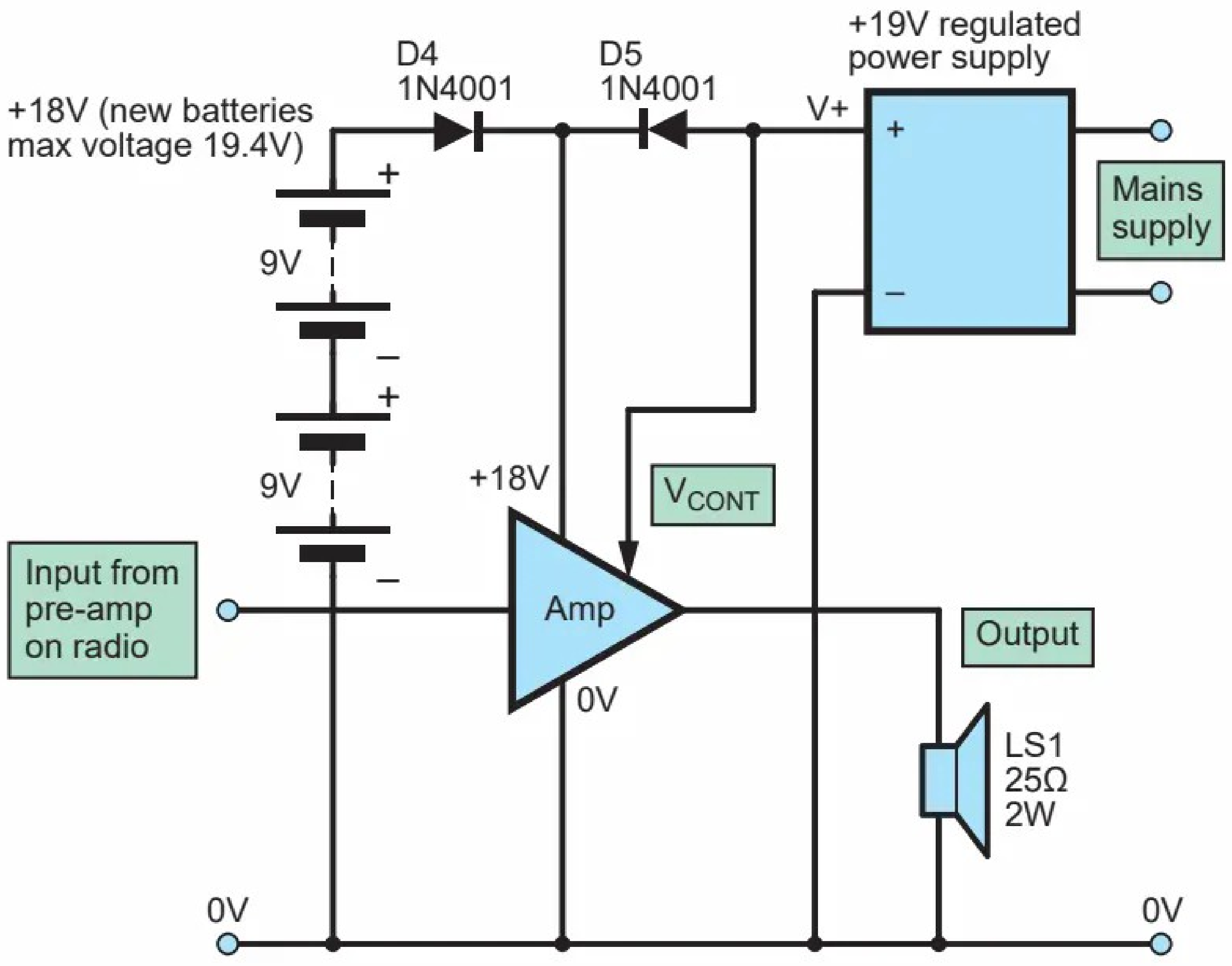
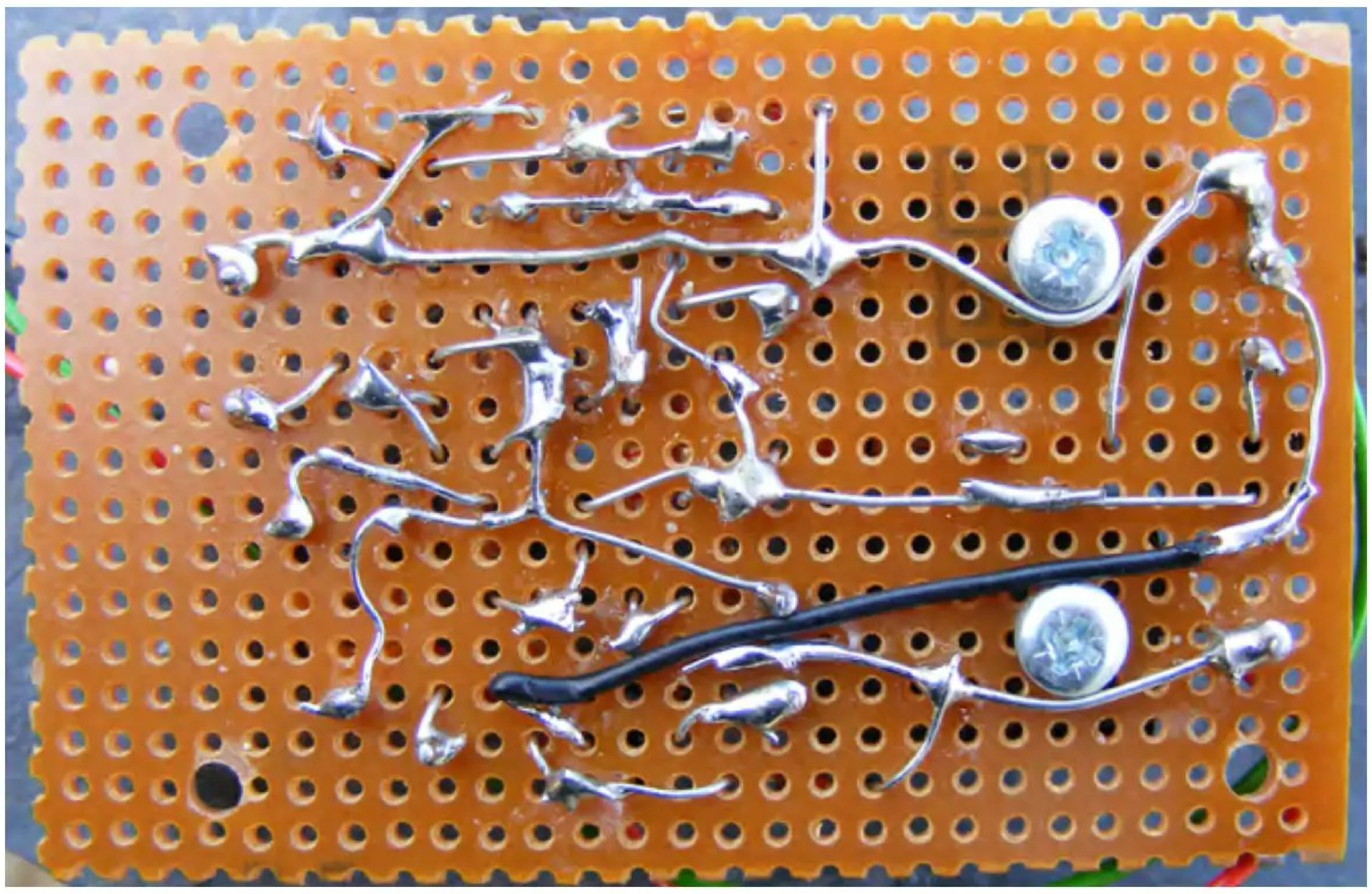


Fig.3: steering diodes can be used to connect the battery, mains supply and amplifier for auto-switching to Class-A when the mains power supply is on.

Photo 4. the underside of the board. This method of hard-wiring was a popular way of working out a PCB layout in the days of designing by soldering iron. ▶



The Fox Report

Barry Fox's technology column

Radio Silence

All around the UK – and the world – rafts of Internet radios with familiar brand names are being silenced because companies you have never heard of have pulled the plug on shadowy but vital enabling services. Set owners can do nothing to salvage the situation; neither can the set makers.

At best, they can helpfully explain the situation; at worst, they will say nothing (in some cases because have gone out of business), leaving owners tearing their hair out, and their radios apart, looking for a non-existent hardware fault.

First thing first. All Internet radios rely on a portal or aggregator service that collates and indexes all the addresses for the Internet radio stations all round the world. These portal services are run by independent specialist companies.

Hardware and software in a radio accesses the index and offers the user a list of stations. The user chooses from the list and the radio goes off into the Internet and accesses the station by its URL.

Because the concept is a bit hard to explain and grasp, this total but invisible reliance on a portal is not publicised; nor is which portal they use. Most radio owners, and often the people selling them, don't care about portals. Until their radio's portal shuts down, anyway.

There is a separate reason why some Internet radios have gone silent. Internet broadcasters such as the BBC have been forced by the ever-increasing numbers of online listeners to stop using the simple Shoutcast Internet radio system.

Shoutcast simply serves HTTP packets from each station to each radio. So a Shoutcast radio can be very simple, with little memory. But Shoutcast cannot handle the huge number of connections now needed. In 2023, the BBC changed to more modern and demanding systems: HLS (Apple's HTTP Live Streaming) and DASH (the Open MPEG standard for Dynamic Adaptive Streaming).

Many receivers have stopped receiving the BBC, but continue to receive other stations – but only for as long as their aggregator portal is supported.

Among the Internet radios affected by the Shoutcast switch-off/portal shutdown is the compact Armour Q2 Cube once sold by QED. These neat little boxes cleverly cut production cost by using a repurposed Bluetooth headset chip.

As well as using Shoutcast, Q2 Cubes relied on the vTuner aggregator/portal service. vTuner has shut down, taking the Cubes with it, along with other brand radios such as older devices from T+A.

Both QED and T+A have now apologised to legacy listeners. Still, QED states, "We have no control over vTuner".

Similar problems arose with the Stream Internet Radios sold by Roberts Radio. These stopped playing and reported a "lost gateway" because they used Reciva chips, which rely on the Reciva portal service. Reciva was originally run by chip giant Qualcomm but is now defunct.

Without a gateway, Reciva-based radios cannot download a software update, and if they have no USB socket, they cannot be updated offline – even if the maker were willing to offer the service.

Roberts offered 20% off the price of a new radio if bought direct from Roberts. This brought the price down to around that of the same radio new from Amazon!

Roberts radios now mostly use a Frontier Smart Technologies chipset (formerly Frontier Silicon, currently doing R&D in Cambridge). But between 2018 and 2020, Smart relied on the vTuner portal. So those owners are likely out of luck.

Luckily for owners of later models, Frontier ditched vTuner in 2020 and switched to another company called Airable, which is based in Germany. A lot of other set-makers are also putting their eggs in the Airable basket.

Internet Radios from Pure, the British company that pioneered DAB, were hit by portal bricking too. Initially, Pure radios used an aggregator system called Flow, which is now dead. Pure now uses the Frontier system with Airable. So owners of post-Flow radios are in luck.

After the demise of Flow and Reciva, many set-makers turned to Californian company TuneIn for its portal service. But music companies Sony and Warner sued TuneIn under copyright law for letting Brits listen to American stations. As a result, in 2020, TuneIn blocked British access to some foreign stations.

Answers to users can be garbled. TuneIn first told listeners of a Californian station that it was blocked because "the broadcaster currently restricts access to this station to their country only". The station angrily responded, "Tune-in's response is totally inaccurate!". TuneIn then changed their tune (ahem) and said, "Because of 'Regional Restrictions', this station is not available in your region".

Airable is capitalising on the TuneIn mess, stating, "Apart from the US Stations you find on products with Airable, and miss on other services, all national and regional BBC stations are available on products with Airable…... These likely are missing on other devices."

Only one thing about Internet radio is truly clear; which aggregator/portal a device uses is now an important factor in deciding whether to buy it.

It's always worth remembering that stations blocked when portals shut down or block zones can still be accessed direct from the station's website or App.

Footnote: Readers with an interest in the history of UK electronics may like to know how the Q2 Cube radio came to be. It was designed for QED by Cambridge Consultants, the lab founded in 1960 by three Cambridge graduates (Tim Eiloart, Rodney Dale and David Southward) who wanted to create clever electronics 'stuff'.

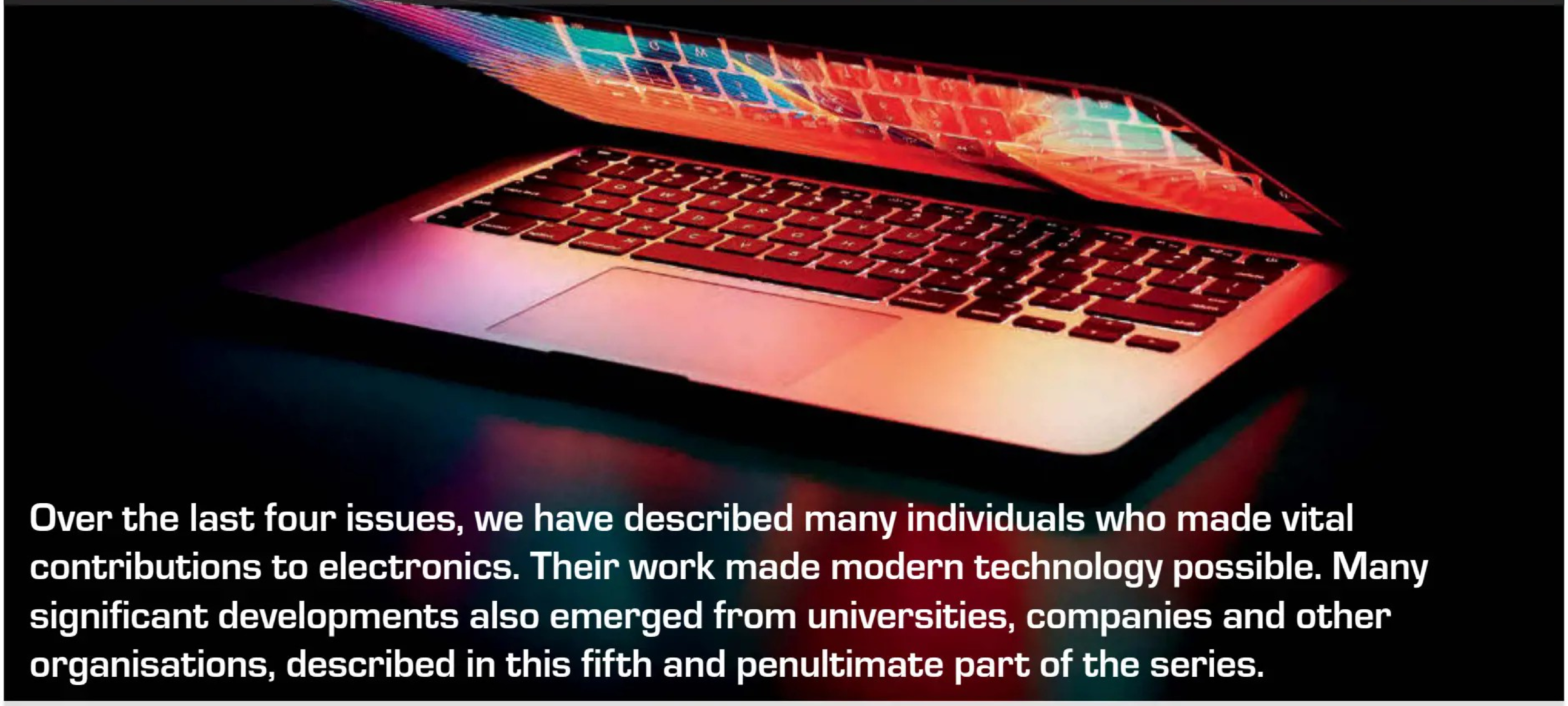
In 1968, CC spawned Cambridge Audio, with circuit designers Gordon Edge and Peter Lee. They twigged that their laboratory power amps might be good for hifi audio and a toroidal transformer could be lighter and smaller than a standard 'box' transformer of the same power.

Their first baby was the legendary P40 (later P50) stereo amplifier, with a hefty toroidal transformer. In the early 1970s, Stan Curtis came on board and widened the range to include top-end CD players and digital decoders with oversampling.

Stan was well known in audio journalism, eg, in *Hi-Fi for Pleasure* magazine, for using solvents to unseal sealed circuit modules and reporting on what circuitry secrets were revealed. He was known in the music biz for digging inside Hammond B3 organs; for instance, to tackle the conductive metal 'whiskers' that grew under the keys and spoiled the sound.

Gordon Edge set up a technology wing for the PA Consulting Group that had been started during WW2. It was based in Melbourn, Herts and designed the ill-fated Strathearn range of stylish hifi gear, which was earmarked for production in Belfast at the height of the 'troubles' – alongside the equally ill-fated DeLorean car, now famous thanks to the movie *Back to the Future*.

The History of Electronics
Inventors and their Inventions



Over the last four issues, we have described many individuals who made vital contributions to electronics. Their work made modern technology possible. Many significant developments also emerged from universities, companies and other organisations, described in this fifth and penultimate part of the series.

Part 5: by Dr David Maddison

Image Source: ***www.pexels.com/photo/2047905/***

This penultimate article covers significant inventions that cannot be attributed to an individual, either because we don't know their name(s) or because they were part of a team. Unlike the last four parts, which were organised by the date of birth of the inventor, we will list them by the year of the invention or discovery. The final article next month will conclude this list.

electric catfish **~2750BCE**

An Ancient Egyptian mural in the tomb of the architect Ti in Saqqara, Egypt refers to electric catfish, later reputed by Pliny and Plutarch to treat arthritis pain and other maladies. This is one of the earliest discoveries involving electricity.

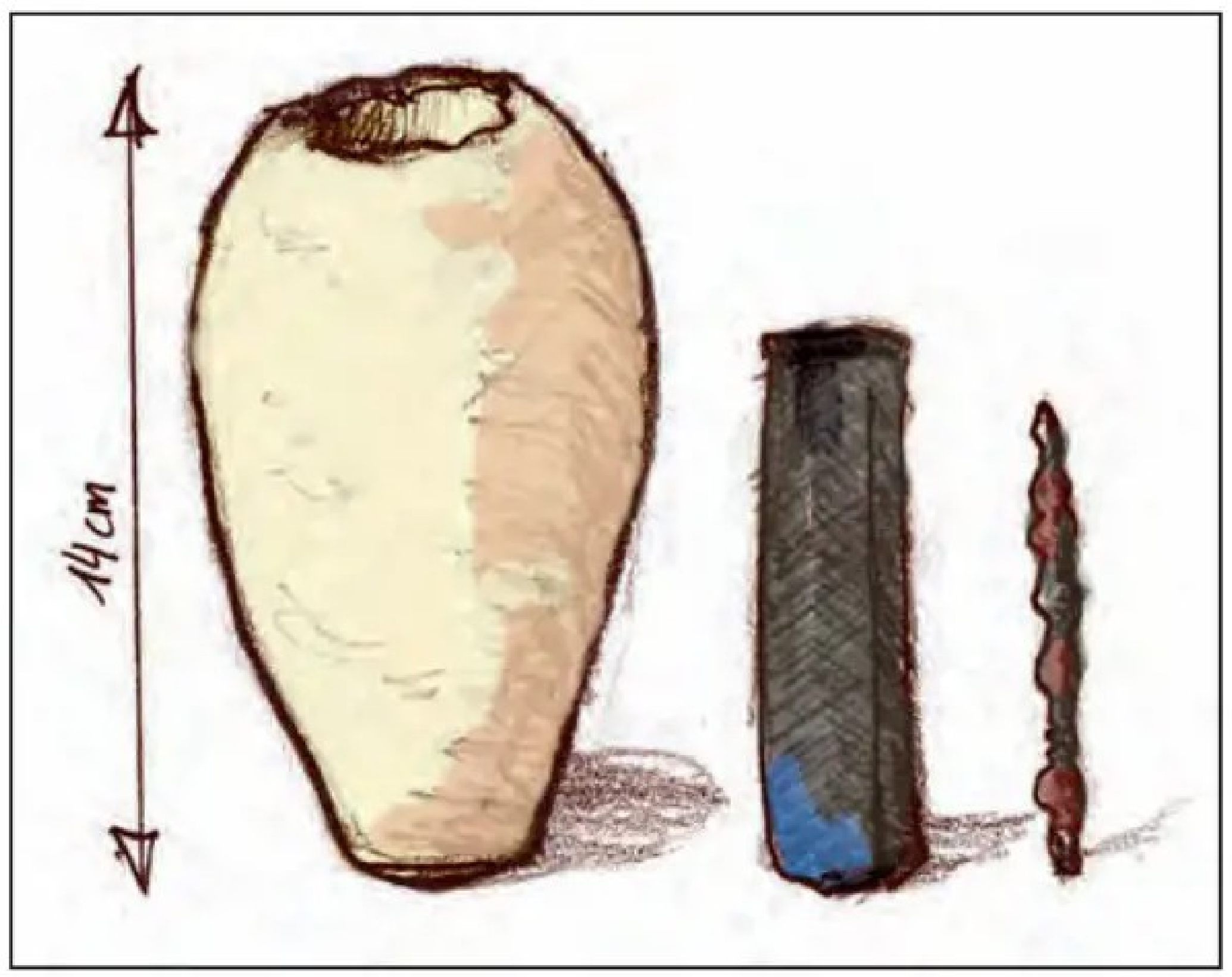


Fig.54: a drawing of the Baghdad Battery. *Source:* https://w.wiki/7FNe

Baghdad Battery **~150BCE – 650CE**

The Baghdad Battery (see Fig.54) is thought to be a battery cell by some, but it could have had other uses and there is no evidence it was used as a battery.

optic fibres **27BCE**

Romans are known to have drawn glass into long flexible fibres, an idea later used for optical fibres for communications and light transmission.

lighthouse, transatlantic cable **1858**

The South Foreland Lighthouse near Dover, UK was the first lighthouse with an electric lamp. It used a carbon arc lamp developed by Frederick Hale Holmes and was on trial until 1860. In 1872, it received a permanent electrical installation.

The light was powered by a pair of coke-fuelled steam engines driving four magnetos, shared with an adjacent lighthouse. Michael Faraday was then investigating electric lighting for lighthouses, and Holmes demonstrated the lamp to him.

The first transatlantic telegraph cable was laid. It worked for only three weeks and took two minutes to transmit one character, about 10 minutes per word.

undersea telegraph cable **1859**

An undersea telegraph cable was laid between Victoria and Tasmania in Australia. It was the longest undersea cable at the time and was retired in 1861.

US transcontinental telegraph **1861**

The US transcontinental telegraph line was completed.

transatlantic cable **1866**

A more functional transatlantic telegraph cable was laid. Messages could be transmitted at eight words per minute.

international telegraph cable **1872**

Australia connected to the international telegraph cable between Darwin and Java.

AU-NZ telegraph link **1876**

The Australia to New Zealand telegraph link was completed.

Australian Overland Telegraph **1877**

The Australian Overland Telegraph line became operational between Port Augusta, SA and Albany, WA, a distance of 3196km.

first international phone call **1881**

The first international phone call was made between New Brunswick, Canada and Maine, USA.

public power station **1882**

The first large public power station, the Holborn Viaduct power station (also known as the Edison Electric Light Station) was built in London. It produced 93kW at 110V DC, with the generator driven by a steam engine. It was preceeded by a small waterwheel-powered generator in Godalming, Surrey that only generated 7.5kW.

The Pearl Street Station opened in New York. It had six 100kW dynamos, was powered by steam and its waste heat was also used for local heating.

| *hydroelectric generation system* **1883**

The Adelaide telephone exchange opened with 48 subscribers, and the Port Adelaide exchange with 21 subscribers.

Australia's first hydroelectric generation system opened at the Mount Bischoff Tin Mine, to power about 50 Swan incandescent lights.

| *Graphophone (phonograph)* **1887**

At the Volta Laboratory (established by A.G. Bell), Chichester A. Bell and Sumner Tainter improved Edison's phonograph by using wax rather than tin foil as the recording medium. Along with Alexander Graham Bell, this confirmed wax as the superior recording medium.

They established the American Graphophone Company to sell their Graphophone product, which was a commercial success.

| *public electricity supply* **1888**

Tamworth, NSW was the first town in Australia with a public electricity supply for arc and incandescent lighting (240V DC).

| *three-phase AC power* **1889**

Young, NSW got three-phase AC power for street lighting, shops, offices and homes.

| *AC hyroelectric power plant* **1891**

The first German three-phase AC power plant started operating in Lauffen am Neckar. 15kV was generated and transmitted 175km to the International Electro-Technical Exhibition in Frankfurt.

Possibly the first commercial AC hydroelectric power plant became operational in Ames, Colorado, USA. It had a capacity of 3.75MW at 3kV, 133Hz, single-phase. That location is still producing electricity but not with the original equipment, although a powerhouse dating from 1905 is still in operation, with a 1904 General Electric generator producing 2.4kV at 1082A (2.6MW).

| *standards for electrical units* **1893**

Standards and definitions for electrical units of **ohms, amps** and **volts** were refined at the International Electrical Congress in Chicago, Illinois, USA.

| *public hydroelectric scheme* **1895**

Australia's first public hydroelectric scheme began operating, to power street lights in Launceston, Tas. In 1921, it was converted to three-phase and 2MW, and was used until 1956.

| *international telegraph* **1902**

Telegraph operations began between Australia and Canada, with connections via Fiji and Norfolk Island.

| *Morse Code transmission* **1906**

The Marconi Company made the first official Morse Code transmission in Australia from Queenscliff, Vic to Devonport, Tas. Some claim Morse radio transmissions were made in 1897 by Professor William Henry Bragg of Adelaide University alone, or with G. W. Selby of Melbourne.

By 1906, Australia had 46 electric power stations with an aggregate capacity of 36MW.

| *production of tungsten-filament bulbs* **1907**

Tokyo Electric Co (predecessor to Toshiba) started small-scale production of tungsten-filament bulbs, reaching full production in 1910.

| *radio broadcasting* **1909**

Radio station KQW started broadcasting in California for experimental, promotional and training purposes by engineer Charles David Herrold (1875-1948). By 1912, he was making scheduled news and music broadcasts (see Fig.55). Many others at the time were only broadcasting Morse Code. He received a commercial license in 1921. The station still exists today as KCBS.

| *amateur radio frequencies* **1912**

The US Government passed the Radio Act of 1912, limiting radio amateurs to frequencies above 1.5MHz, as those frequencies were considered useless. This led radio amateurs to discover HF radio propagation via the ionosphere. In 1921, a one-way transmission was made across the Atlantic; then, in 1923, two-way transmission (see https://pemag.au/link/abnv).

| *transcontinental phone call* **1915**

The first transcontinental phone call was made in the USA, over 5794km, facilitated by the newly-invented vacuum tube amplifier.

| *rotary dial telephones* **1919**

Bell System, USA made the first rotary dial telephones.

| *commercial radio* **1920**

The world's first commercial licensed radio station, KDKA in Pittsburgh, Pennsylvania, USA started broadcasting.

| *double-coiled tungsten filament* **1921**

At Tokyo Electric Co, Junichi Miura made the first double-coiled tungsten filament light globe using the technique developed by Benbow (1917).

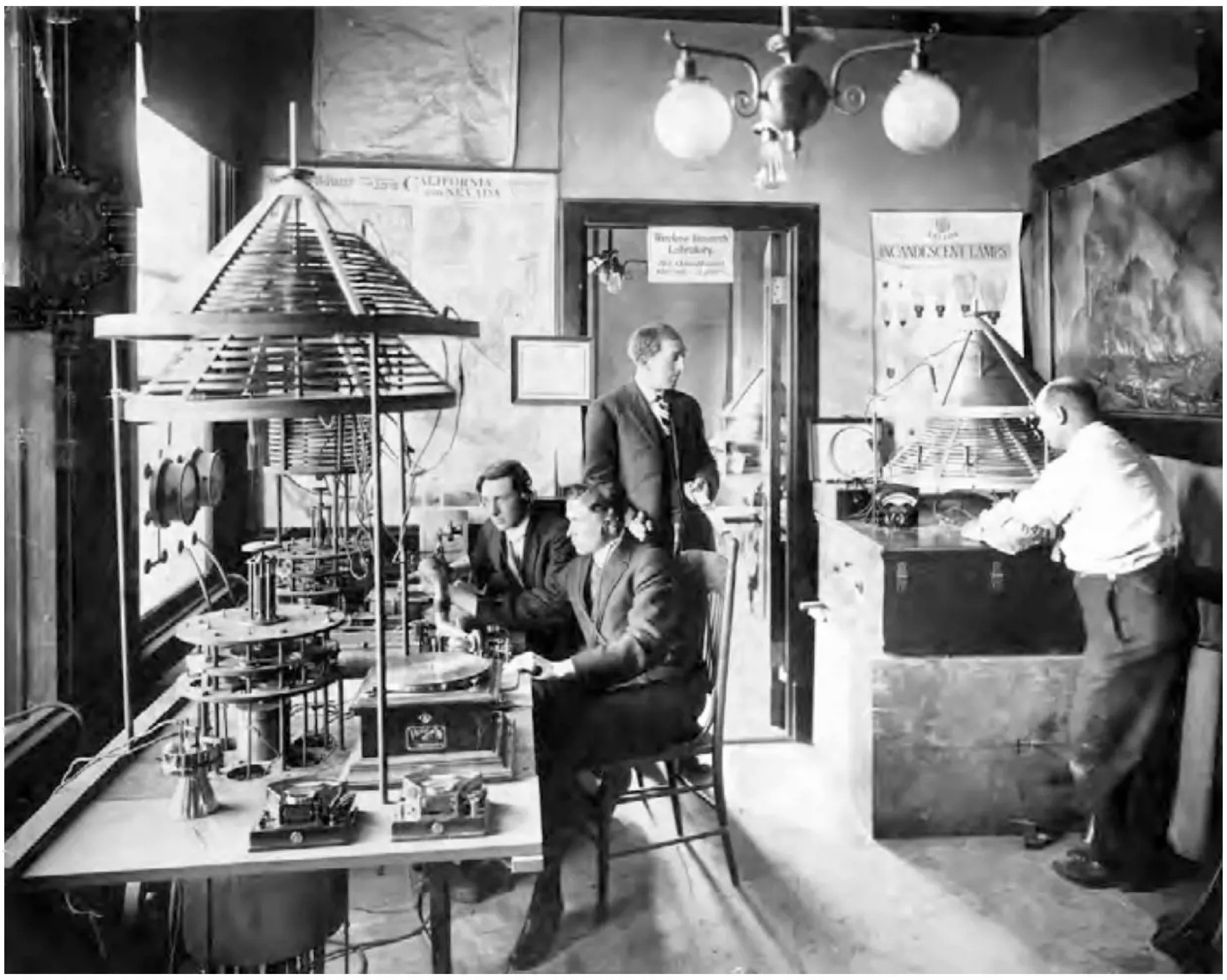


Fig.55: Charles Herrold's San Jose, California radio laboratory, circa 1912. He transmitted from this location as radio KQW. Herrold is standing in the doorway. *Source:* https://w.wiki/7EFw

It entered small-scale production in 1930 and mass production in 1936.

“mobile phone” **1922**

Early experiments were conducted with a “mobile phone”. The phone was a portable two-way radio that used an umbrella antenna with a fire hydrant for its Earth. Music was transmitted from a base station to the radio. See the YouTube video titled “World’s First Mobile Phone (1922)” – https://youtu.be/lLiLaRXHUr0

transatlantic telephone call **1926**

The first transatlantic telephone call was made.

car radio, Phonovision **1927**

The first mass-produced car radio was made, the Philco Transitone. Before that, radios were adapted for car use. The exact year is subject to some dispute.

John Logie Baird made the first “videodisc” player as a proof-of-concept called the Phonovision. The output of mechanical television scanning from a Nipkow disc was recorded to a gramophone record. It only had a 30-line resolution at 5FPS. Some recordings were found, and in 1982-87, software was made to recover the images.

See the website at https://pemag.au/link/abnw and the video titled “30-line TV video recordings news feature”, plus other videos below:

https://youtu.be/J2mb4R9W9TI
https://pemag.au/link/abnx
https://youtu.be/G3CFkK5OORw

blind aircraft take-off and landing **1929**

The first blind aircraft take-off and landing was made by Lt James Doolittle in a Consolidated NY-2 biplane. It was instrumented with a Kollsman altimeter, Sperry directional gyroscope and an artificial horizon, with a radio range and marker beacon by the National Bureau of Standards and a special radio receiver with a vibrating reed display by Radio Frequency Laboratories.

synthesis of ferrite compounds **1930**

Yogoro Kato and Takeshi Takei at the Tokyo Institute of Technology first synthesised ferrite compounds. These materials are used in inductors, transformers and electromagnets, electrical noise control, early computer memories, magnetic tapes, radar absorbing materials, loudspeakers and magnets.

TDK incorporated the first ferrite cores in radios in 1937, making them smaller and lighter. TDK was the only company that could supply ferrite cores until the end of WW2.

LP records **1931**

RCA introduced the first commercial LP (long play) records. They were 12 inches or 30cm in diameter, ran at 33⅓RPM and contained up to 11 minutes of audio per side (the same time as a standard 1000ft/305m movie reel). They were a commercial failure due to the expense of playback equipment and the Great Depression.

Magnetophon K1 tape recorder **1935**

German company AEG introduced the Magnetophon K1, the first practical tape recorder (see Fig.56). It used iron-oxide-coated non-metallic magnetic tape.

The tape was originally based on Fritz Pfleumer’s idea (see his entry on p58 of the March issue), with further development by Friedrich Matthias. A non-damaging head was designed by Eduard Schüller, who also built the prototype machines.

The audio quality was poor until Walter Weber (1907-1944) discovered the AC biasing technique (by accident!), dramatically improving audio quality. These recorders had all the basic features that were incorporated into later analog tape recorders. You can see a video on the similar Magnetophon FT4 at https://youtu.be/cLjD0B6QoaM

high-definition television **1939**

German company Fernseh AG demonstrated high-definition 1029-line TV for displaying military maps. This system required 15MHz of bandwidth, which is why HDTV wasn’t widely introduced until the advent of digital broadcasting in the 1990s.

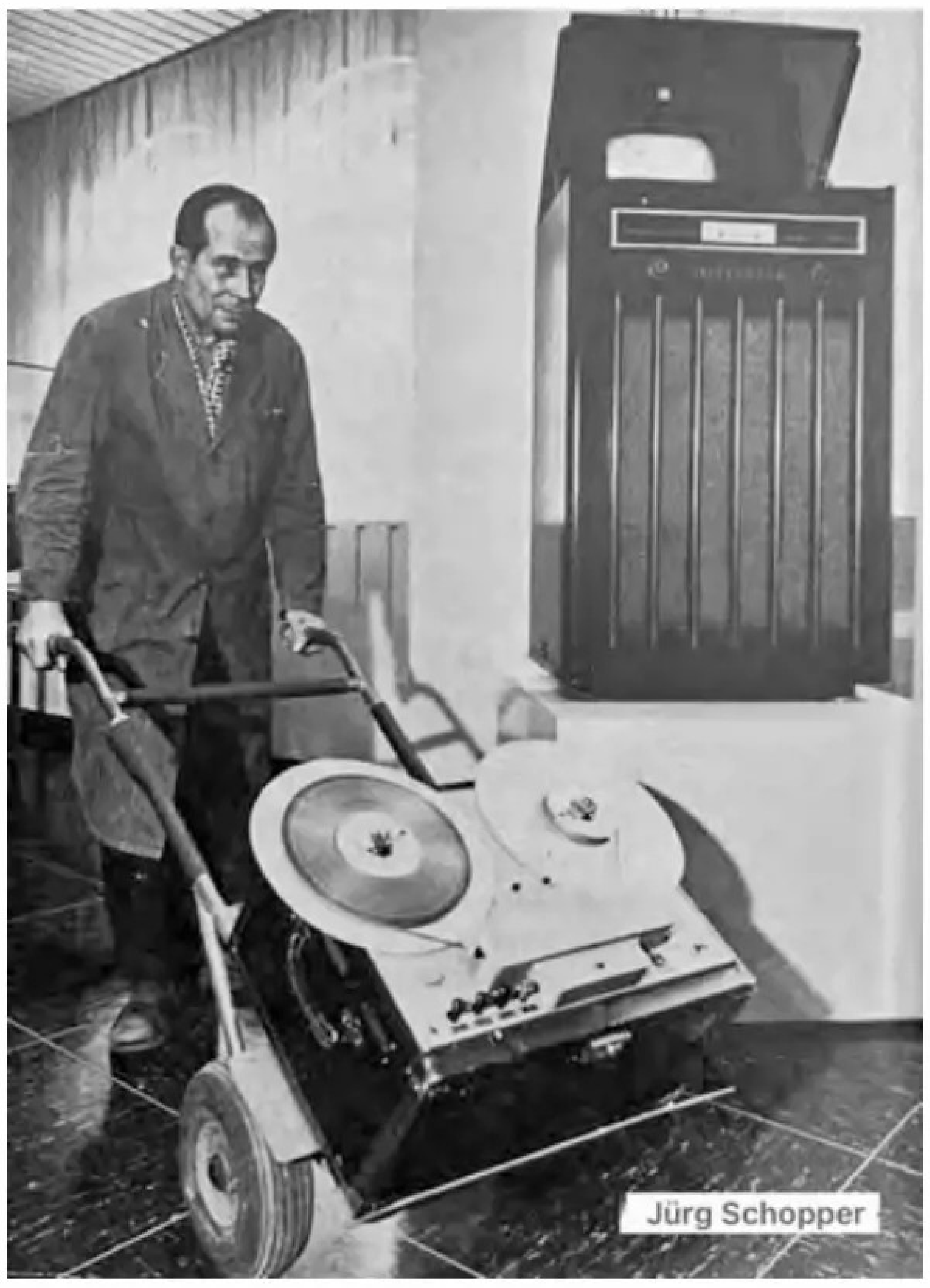


Fig.56: the AEG Magnetophon K1 tape recorder being delivered to the Berlin Radio Show in 1935. *Source:* https://museumofmagneticsoundrecording.org/ManufacturersAEGMagnetophon.html

commercial FM broadcasting, NTSC **1941**

Commercial FM broadcasting formally began in the USA, although there were experimental transmissions before that. It was on the 42-50MHz band, split into 40 channels. In 1945, it was reassigned to 88-106MHz band with 80 channels, then extended to 108MHz and 100 channels.

The monochrome NTSC television standard was released.

“Colossus” digital computer **1943**

The first programmable digital computer was built, the British “Colossus”.

“ENIAC” digital computer **1945**

The US “ENIAC” computer was built, the world’s first general-purpose programmable digital computer.

The electronic Merrill Wheel-Balancing System for cars was also invented.

FM broadcasting in Australia **1947**

Experimental FM broadcasting in Australia took place from 1947 to 1961 but with an extremely limited audience (the receivers were costly). It was discontinued to clear the TV band and eventually reintroduced on a band that no one else in the world used. Fortunately, in 1975 it was reintroduced on the widely used 88MHz-108MHz.

vinyl records **1948**

Columbia Records used PVC to make vinyl records, which are more durable than previous shellac compounds. They could be made with much finer grooves called “microgrooves”. These allowed for a playback time of about 22 minutes on a 12in/30cm disc (there was also a 10in/25cm disc). Peter Carl Goldmark (1906-1977) developed the format.

45RPM records **1949**

Columbia competitor RCA introduced the 45RPM record with a 7in/18cm diameter, intended as a replacement format for 78RPM records, with a similar duration of about five minutes per side. Eventually, “quality music” was distributed on 33⅓RPM records, with “popular music” on 45RPM records. Both formats are still around today.

permanent magnets, PCBs **1950**

Philips accidentally discovered barium hexaferrite, which became a popular permanent magnet material.

The History of Electronics, part 5

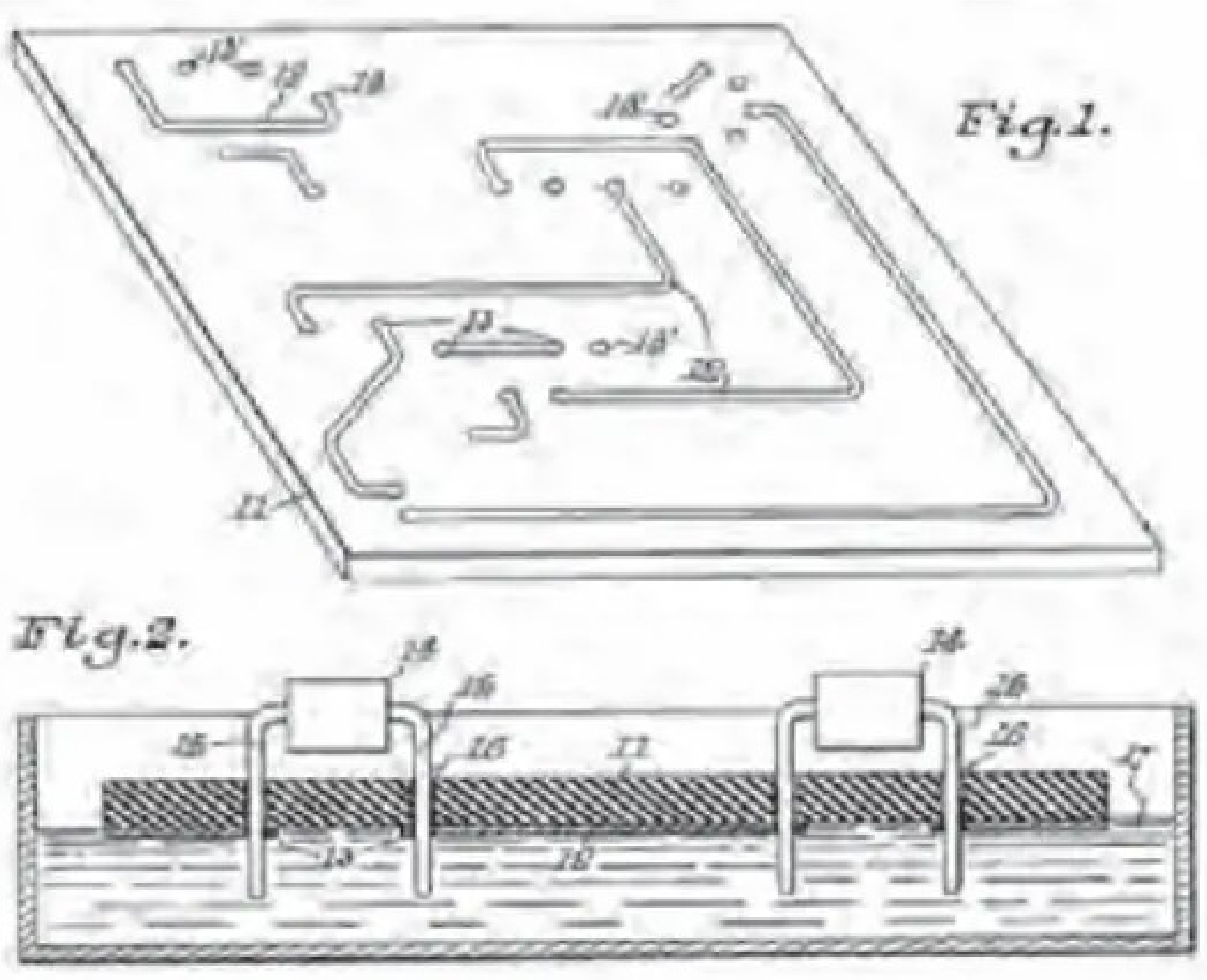


Fig.57 (above): an image from the 1956 US Army patent 2,756,485 for PCB manufacturing.

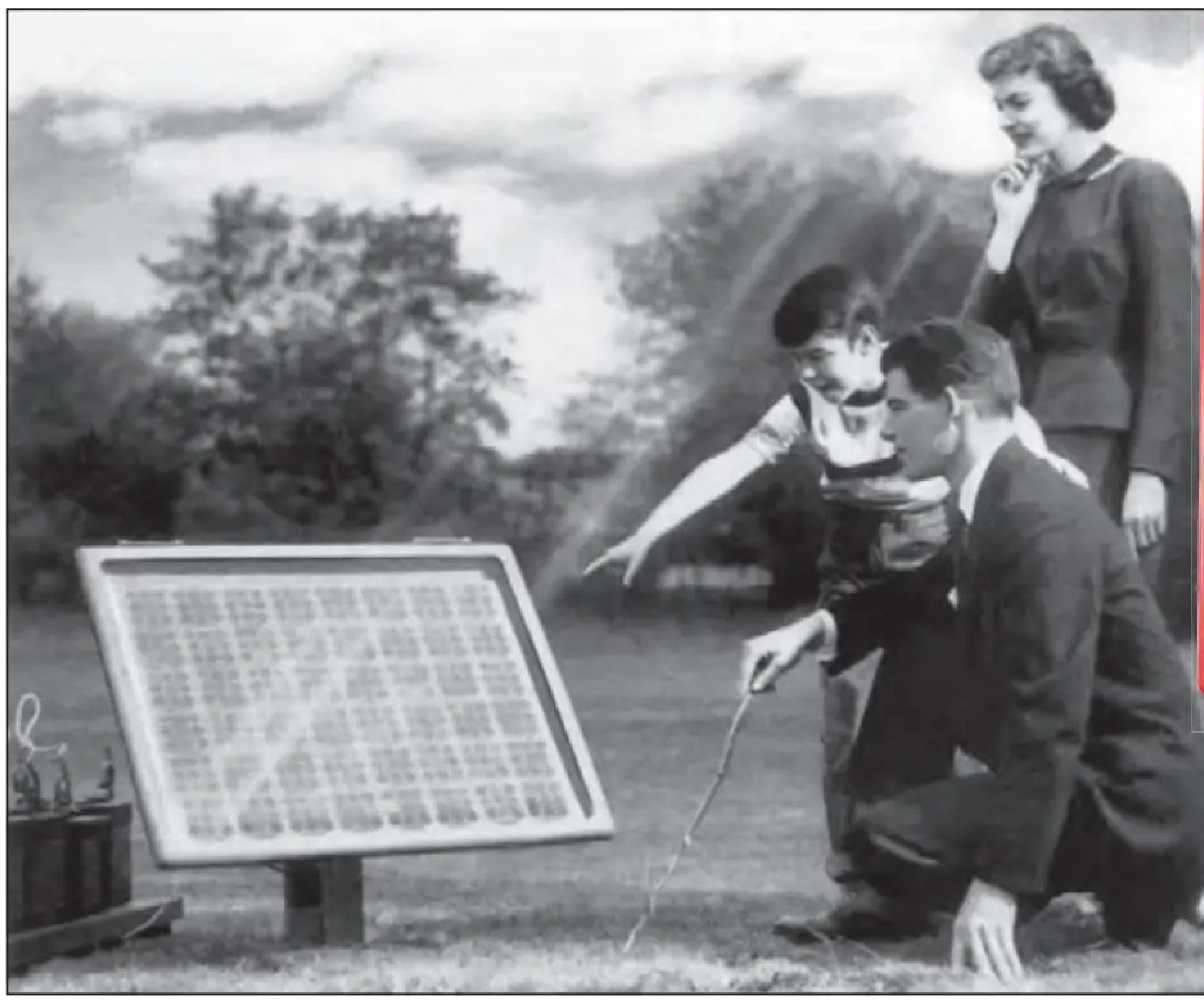


Fig.58 (right): an advertisement for the first practical solar cell by Bell from the 25th of April, 1954. It had a 6% efficiency. *Source: www.onthisday.com/photos/1st-solar-battery*

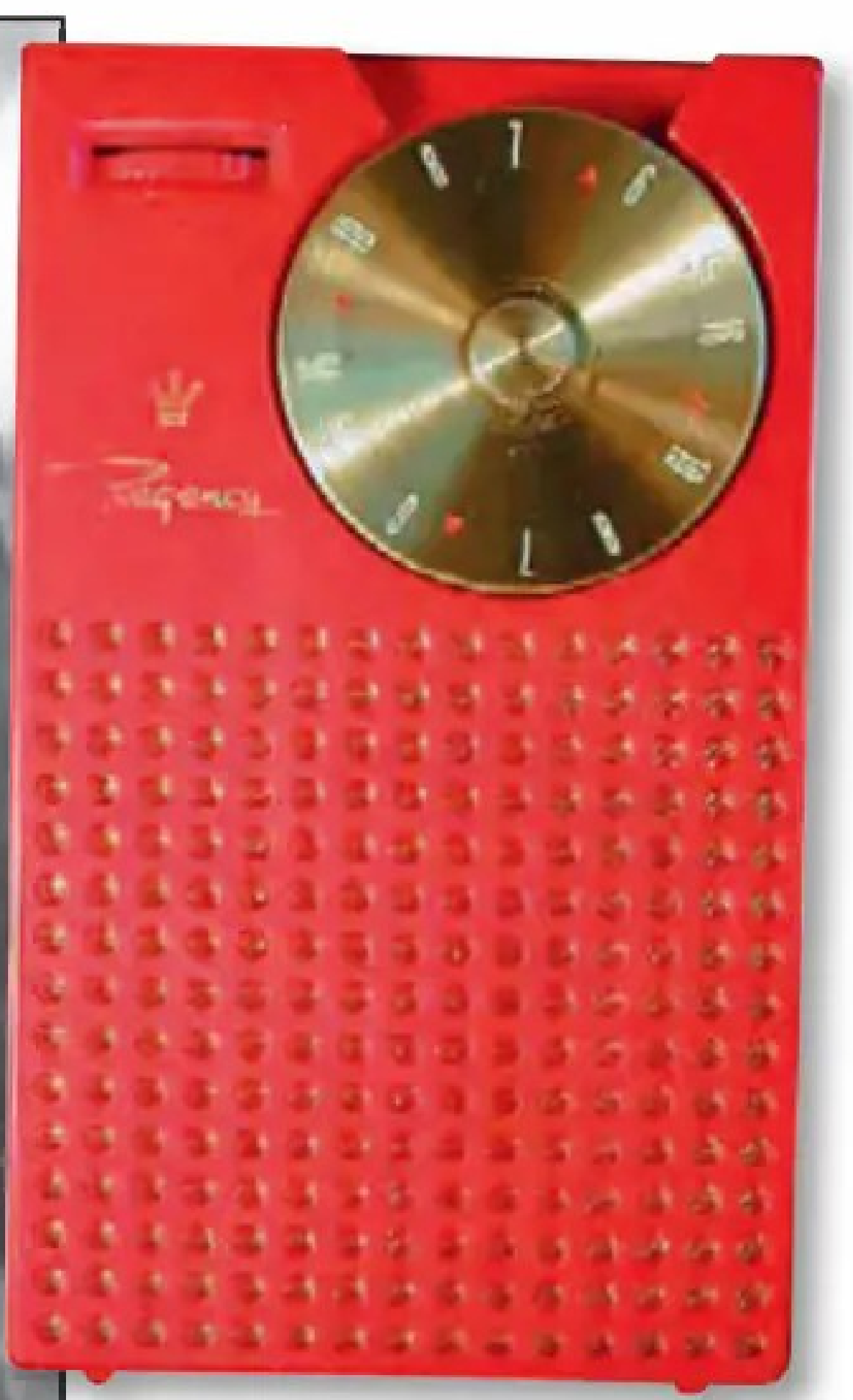


Fig.59 (above): the Regency TR-1, the first commercial portable transistor radio.

The US Army applied for US patent 2,756,485, granted in 1956, titled "Process of Assembling Electrical Circuits" (see Fig.57). This led to the mass production of printed circuit boards.

nuclear power, colour TV etc **1951**

Sony released the H-1 magnetic audio tape recorder for consumer use. It was the first tape recorder designed for domestic use and weighed 13kg.

The first nuclear power reactor (EBR-1) became operational in Arco, Idaho, USA. It could power four 200W light globes.

CBS in the USA demonstrated colour TV broadcast using the electromechanical field-sequential system (FSC) standard. There were very few appropriate receivers. That standard was withdrawn and the NTSC standard was subsequently used instead.

speech recognition system, video game **1952**

The first speech recognition system was demonstrated, which could recognise one speaker saying the digits zero to nine with 90% accuracy. It was called Audrey (Automatic Digit Recognition machine) and was produced by H.K. Davis at Bell Laboratories.

The first computer game was created by Alexander Shafto Douglas (1921-2010) at the University of Cambridge in England. It was called "OXO" and was a version of noughts and crosses (also known as tic-tac-toe).

maser, nuclear submarine, NTSC **1953**

The first **maser** (microwave laser; microwave amplification by stimlated emission of radiation) was built by Charles H. Townes, James P. Gordon, and Herbert J. Zeiger at Columbia University. Masers are used as highly-stable frequency references and extremely low-noise amplifiers for microwave frequencies. They can also generate electromagnetic waves at microwave and other frequencies.

The first nuclear-powered submarine, the USS Nautilus, was launched.

The NTSC colour TV standard was released.

photovoltaic cell, transistor computer **1954**

The first practical photovoltaic cell was developed at Bell Laboratories (see Fig.58) by Calvin Souther Fuller (1902-1994), Daryl Chapin (1906-1995) and Gerald Pearson (1905-1987).

The world's first practical colour television broadcast (NTSC) began in the USA. However, most programming remained in monochrome for some time due to the high cost of sets and lack of colour source material.

TRADIC (for TRAnsistor DIgital Computer or TRansistorized Airborne DIgital Computer) was the world's first fully-transistorised computer, built by Bell Labs for the US Air Force. It included 684 Bell 1734 Type A point contact cartridge transistors and 10,358 germanium diodes.

The first transistorised portable radio went on sale, the Regency TR-1. It used four Texas Instruments NPN transistors, a 22.5V battery and cost US$49.95, equivalent to about $850 today (about what collectors pay!). It is shown in Fig.59 and was described in Silicon Chip magazine, April 2013 (https://siliconchip.au/Article/3761).

programmable music synthesiser etc **1955**

The RCA Mark I Synthesiser was the first programmable music synthesiser. There is an interesting article about how it works at https://pemag.au/link/abny

The first wireless TV remote control was introduced, the Zenith Flash-matic. It used visible light and had to be directed at one of four photocells in each corner of the screen to perform various functions (on/off, mute or change channel).

IBM 350 drive, VRX-1000 recorder etc **1956**

The first commercial disk drive, the IBM 350 (Fig.60), went on sale. It stored 3.75MB and weighed about one tonne.

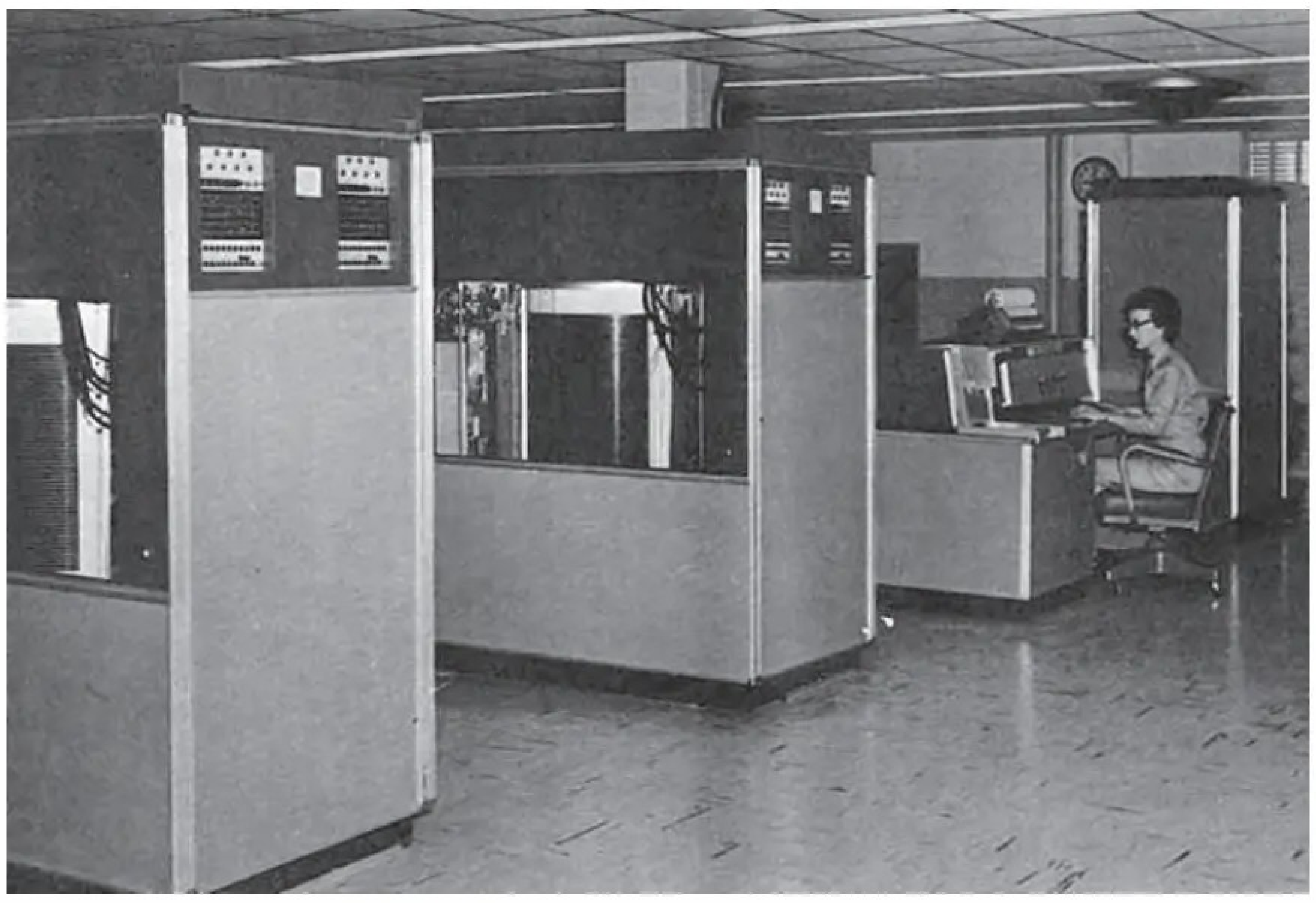


Fig.60: two IBM 350 disk drives at the US Army Red River Arsenal. *Source: https://w.wiki/7EFy*

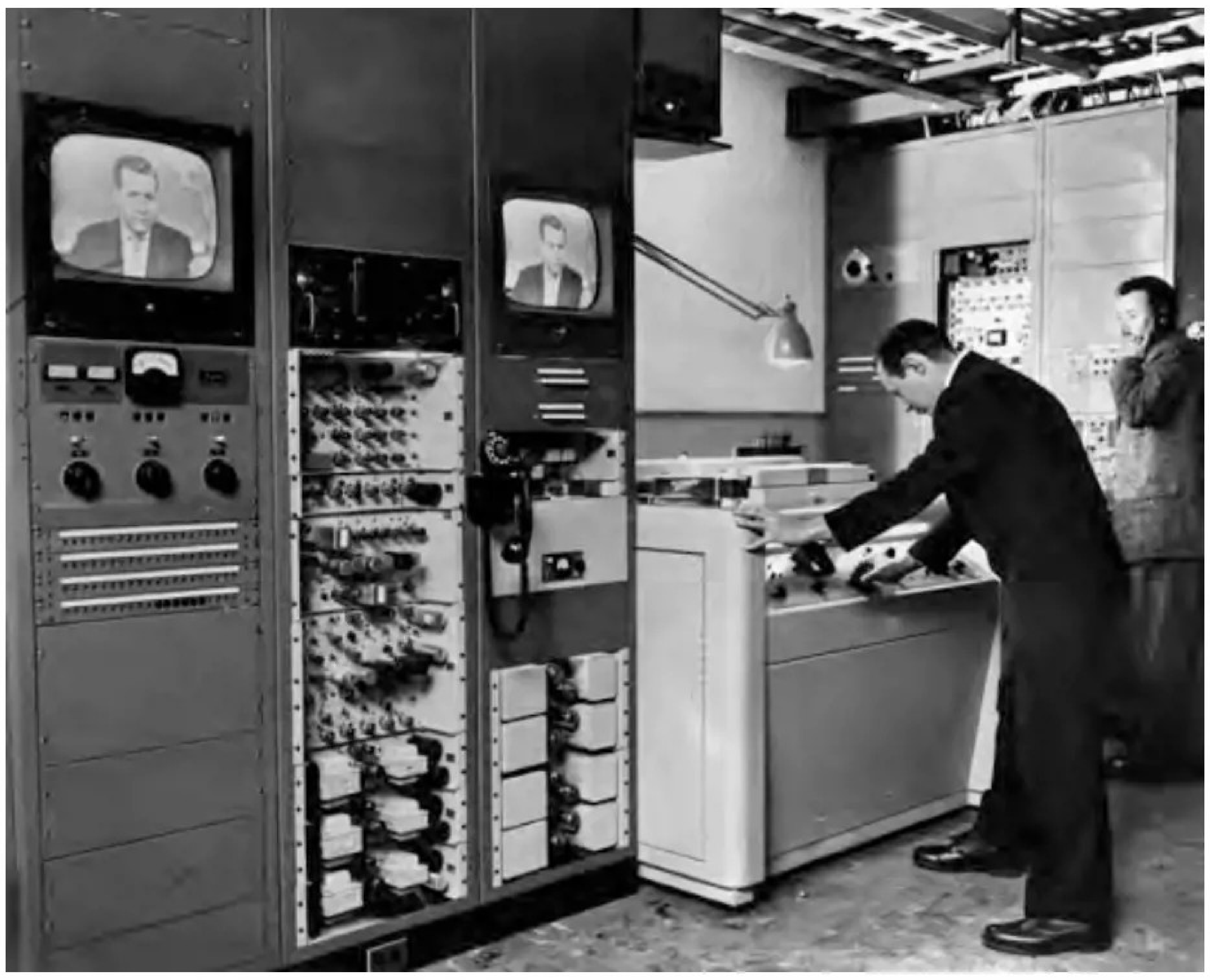


Fig.61: CBS engineer John Radis operating an Ampex VRX-1000 videotape recorder on the 30th of November, 1956. It was the first time this machine was used on a broadcast program. *Source: www.quadvideotapegroup.com/2015/12/*

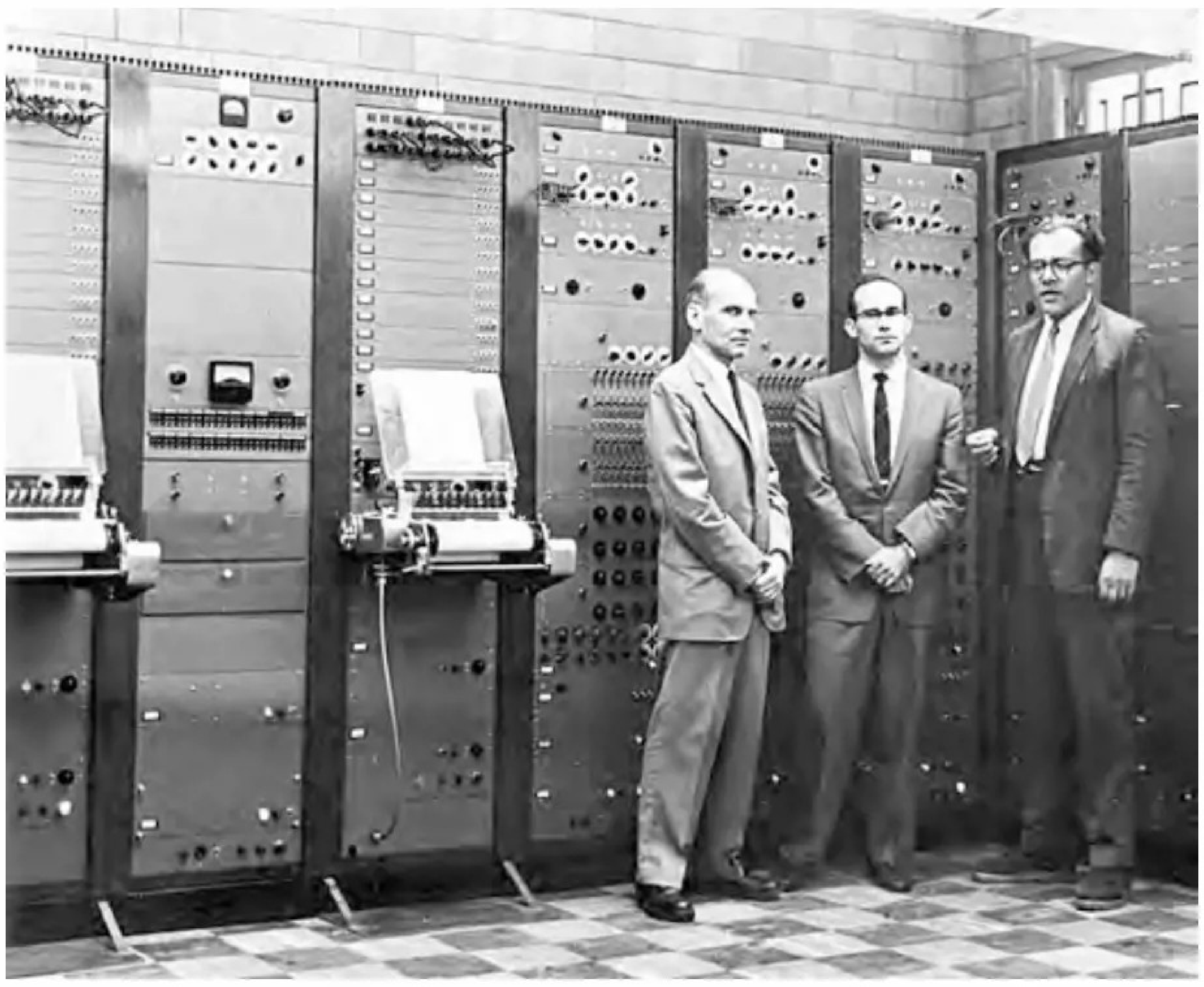


Fig.62: the RCA Mark II synthesiser. Note the punch paper terminals. *Source: https://electronicmusic.fandom.com/wiki/RCA_Synthesizer (CC-BY-SA).*

The first commercial video tape recorder, the monochrome VRX-1000, was introduced by Ampex for studio use (see Fig.61). It used two-inch (5.08cm) wide tape in the Quadruplex format. It cost US$50,000, equivalent to about $840,000 today.

The machine's major innovation was transverse recording, where the video image was written across the tape rather than linearly, allowing for a reasonable tape speed of 38cm per second. Before its introduction, the only way to record TV programs was with film. Silicon Chip published a detailed article on Quadruplex recording in its March 2021 issue (**https://siliconchip.au/Article/14782**).

The first transatlantic telephone cable was laid, TAT-1 (Transatlantic No. 1). It could carry 35 simultaneous telephone calls with a 36th channel that carried 22 telegraph circuits.

R-7 ICBM, Sputnik 1 satellite etc **1957**

The RCA Mark II synthesiser (Fig.62) was a successor to the Mark I and much easier to program. It had two punch paper terminals for playing compositions. These stored data for playback; the machine's output was recorded on lacquer-coated record-like discs. See:

https://youtu.be/rgN_VzEIZ1I
https://pemag.au/link/abnz

The world's first large-scale civilian nuclear power plant began operation in Shippingport, Pennsylvania, USA. Its primary purpose was to produce electricity, but it was also a proof-of-concept of the breeder reactor, where the reactor transmutes non-fissile fuel into fissile fuel at the same time as producing power. It remained in operation until 1982.

The first ICBM (intercontinental ballistic missile), the Soviet R-7 Semyorka, was introduced. ICBMs were later reused by multiple nations as rockets for launching satellites and other space missions.

The FORTRAN computer language was commercially released. It is still used by mathematicians, scientists and engineers.

The Soviet Union launched the first artificial satellite, Sputnik 1, using an R-7 Semyorka rocket. Silicon Chip had a detailed article on Sputnik's radio transmitters in its November & December 2023 issues (**https://siliconchip.au/Series/407**).

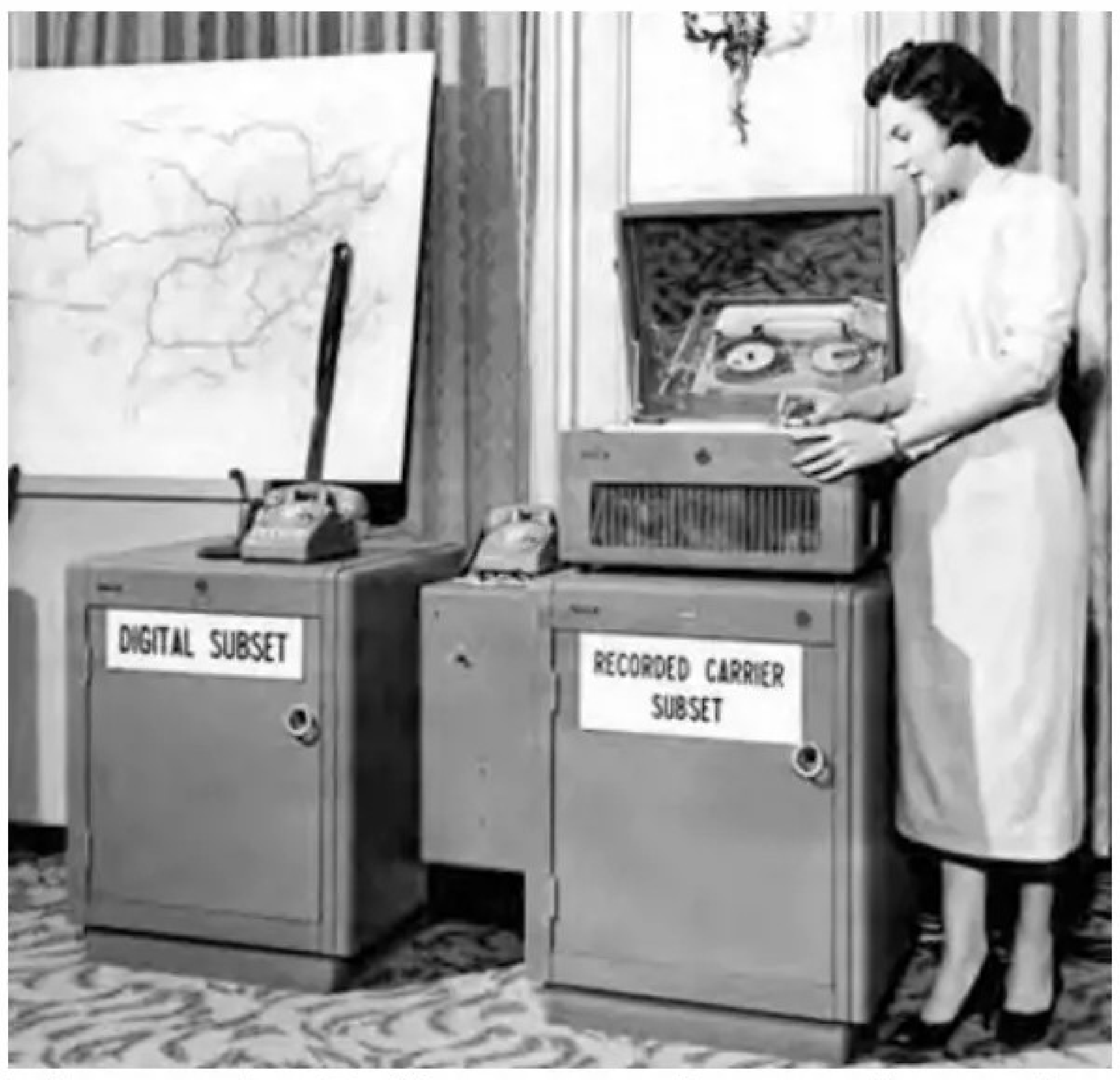


Fig.63: the Bell 101 modem, released by AT&T in 1958. *Source: https://history-computer.com/modem-complete-history-of-the-modem/*

colour videotape, modems, pacemaker **1958**

The first US satellite, Explorer 1, was launched in 1958 on a Juno I rocket booster.

The Ampex VR-1000B commercial colour videotape recorder was released. It supported multiple international video standards. You can see the product brochure at **https://pemag.au/link/abp0**

Telephone-line modems (modulators/demodulators) were mass-produced for the military in the USA as the Bell 101 modem in 1958 (Fig.63), used for the SAGE air defence system. The technology was made available to the public in 1959, with a 110bit/s speed. They were a development of the teleprinter multiplexers used by news services and the like in the 1920s.

The first implantable cardiac pacemaker was released.

The second ever computer game was created by William Higinbotham (1910-1994) at Brookhaven National Laboratory, New York, USA. It was called Tennis for Two and was similar to Pong.

Veroboard, Mosfet, planar process etc **1959**

The first American ICBM, the SM-65 Atlas, went into operation. It was also used to launch Project Mercury astronauts.

What was to become Veroboard for electronics prototyping and one-off circuits was invented.

The first commercial plain-paper photocopier, the Xerox 914, was introduced. See the video at **https://youtu.be/9xZYcWsh8t0**

The History of Electronics, part 5

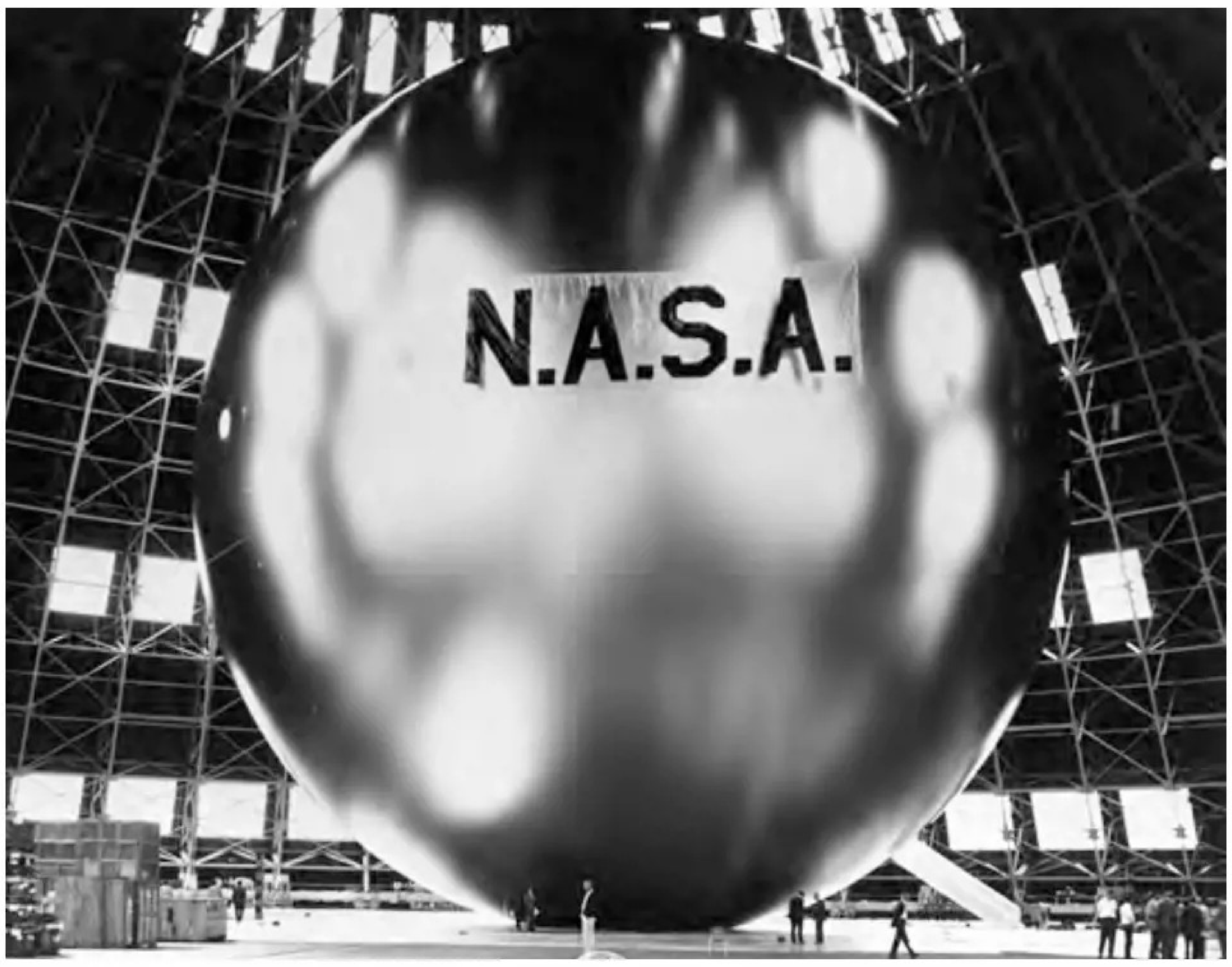


Fig.64: the ECHO 2 satellite undergoing testing and inspection, dwarfing the people around it. The first transmission using ECHO was from California to New Jersey in 1960. *Source: NASA.*

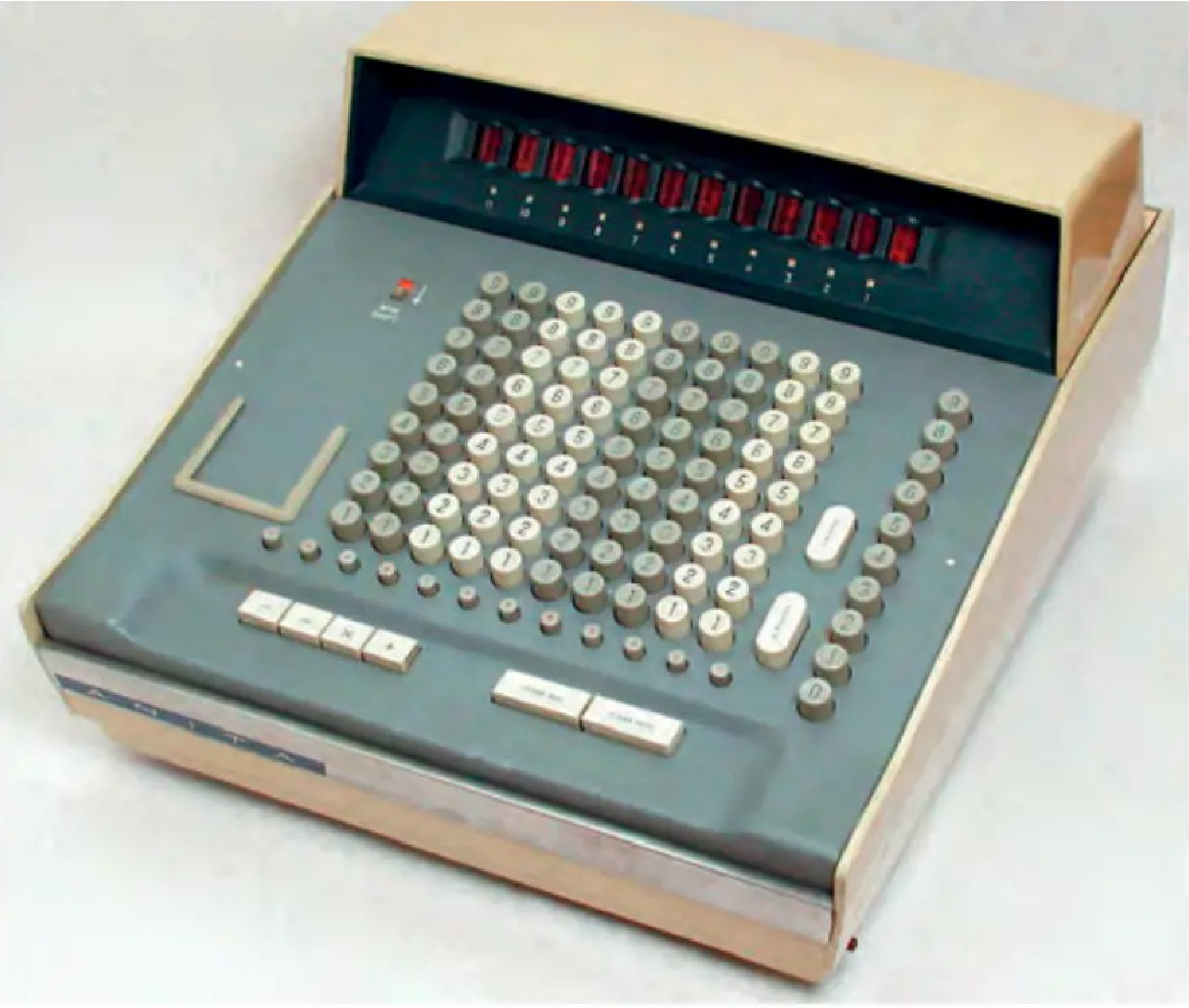


Fig.65: the Anita Mk VII & VIII (pictured) were launched simultaneously in 1961. VII was the first model because they had used the previous numbers for their mechanical calculators *Source: https://w.wiki/7EFz (GNU FDL).*

Mohamed Atalla and Dawon Kahng at Bell Laboratories invented the Mosfet (metal-oxide-semiconductor field effect transistor). This was described in an article in Silicon Chip magazine, May 2022, on the development of transistors (**https://siliconchip.au/Article/15305**).

The semiconductor planar process for fabricating integrated circuits was invented by Jean Amédée Hoerni (1924-1997). Silicon Chip published a series on the history of ICs in June-August 2022 (**https://siliconchip.au/Series/382**).

NASA's Project Echo, SMT parts **1960**

NASA started Project Echo. Echo 1 and Echo 2 (launched 1964) were experimental passive reflector communications satellites (Fig.64), 30m diameter inflated balloons with some electronics onboard. They provided valuable data about atmospheric drag and other information.

IBM first demonstrated surface-mounting component technology (SMT) in a small computer. It was later applied to the Launch Vehicle Digital Computer in the Saturn IB and Saturn V in the 1960s.

ANITA electronic calculators, LEDs **1961**

The first electronic calculators were the ANITA Mark VII and Mark VIII, released in 1961, using vacuum tubes and cold cathode tubes (see Fig.65). The first solid-state electronic calculator was the Friden EC-130, released in 1963.

J. W. Allen and R. J. Cherry invented the first visible light LEDs at SERL in Baldock, UK.

Josephson junction, Telstar 1 etc **1962**

The third computer game was invented by Steve Russell (1937~) and played on a Digital PDP-1 mainframe computer. It was called Spacewar! – see Fig.66 and the video titled "Spacewar! (1961) - First digital computer game":

https://youtu.be/CwZAKJ8Y6YU

The Josephson effect was observed in this year but was not recognised for what it was at the time. It eventually led to a superconducting circuit called the Josephson junction, with applications in quantum computers, voltage standards and digital signal processing, among others. It was named after Brian David Josephson (1940~).

The IBM Shoebox speech recognition system could recognise 16 spoken words (numerals and arithmetic operators). It was a voice-operated printing calculator (see Fig.67).

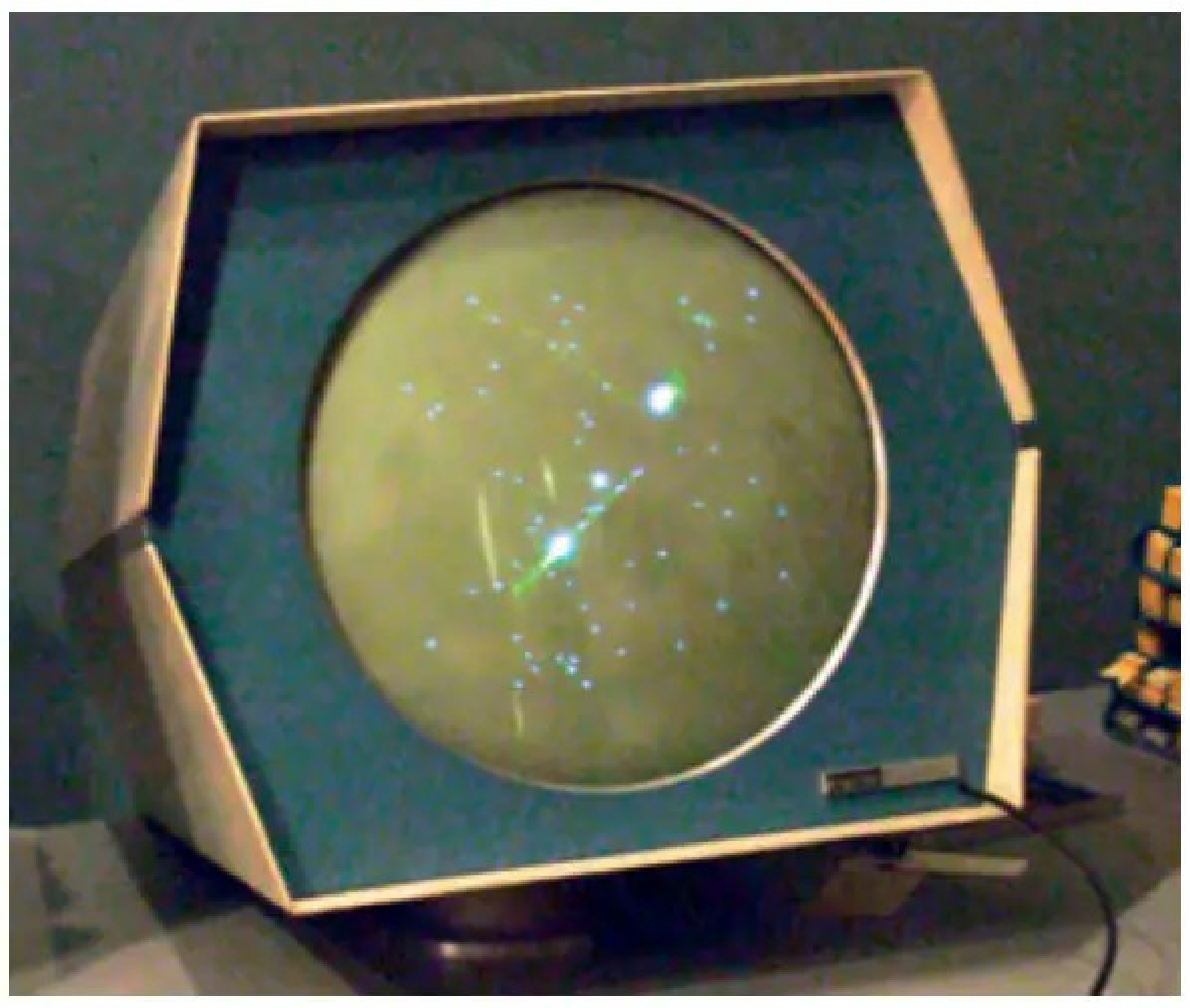


Fig.66: Spacewar! Running on a PDP-1 computer. *Source: https://w.wiki/7EF$ (CC-BY-2.0).*

The Telstar 1 communications satellite was launched into an elliptical orbit (not geostationary). Telstar 2 was launched in 1963. Both satellites were experimental; neither are still in use but are in orbit. Telstar 1 carried the first transatlantic TV transmission via satellite that same year; data was also transmitted between two IBM 1401 computers via Telstar 1.

Conclusion next month

In the next issue, we will pick up this list of discoveries by companies and organisations in 1963. That will be the final instalment in this six-part series on the history of early electronics.

The developments covered in that last article will include undersea data cables, electronic calculators, fax machines, video casette recorders, computers, mobile phones and more. **PE**

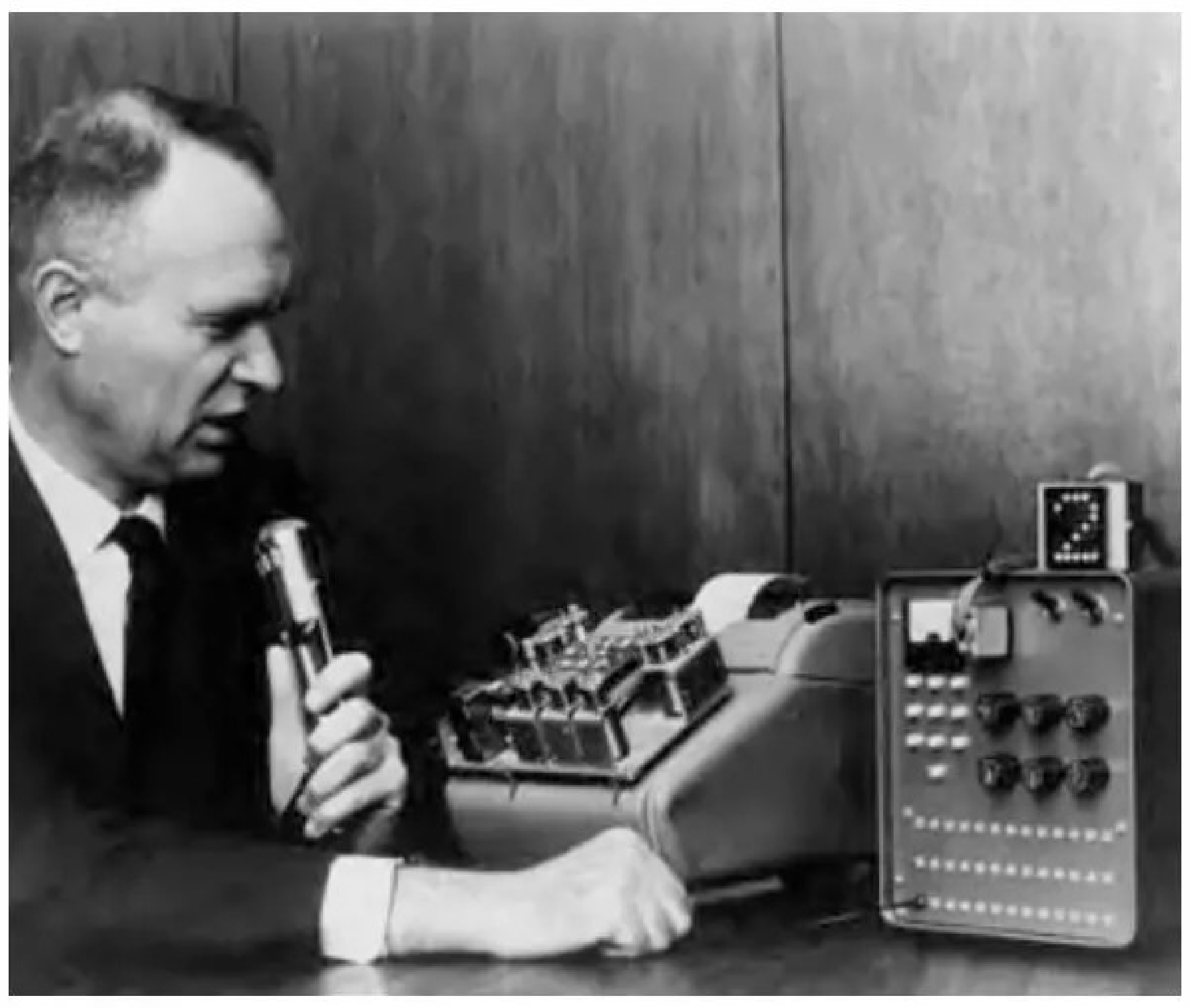


Fig.67: an IBM 'Shoebox' voice recognition system/calculator. *Source: IBM.*

Practical Electronics PCB SERVICE

PROJECT	CODE	PRICE
MAY 2025		
Skill Tester 9000	08101241	£9.99
USB-C Serial Adaptor (black solder mask)	24106241	£2.99
DC Supply Protector (SMD adjustable version)	08106241	£3.99
DC Supply Protector (TH adjustable version)	08106242	£3.99
DC Supply Protector (TH fixed version)	08106243	£3.99
APRIL 2025		
ESR Test Tweezers PCB set	04105241+	£5.99
Fan Speed Controller Mk2	10104241	£4.99
Calibrated Microphone (SMD version)	01108231	£3.99
Calibrated Microphone (TH version)	01108232	£3.99
MARCH 2025		
Pico Gamer	08104241	£9.99
Compact Frequency Divider	04112231	£4.99
FEBRUARY 2025		
Laser Communicator transmitter	16102241	£3.99
Laser Communicator receiver	16102242	£3.99
Model Railway Points Controller transmitter	09101241	£3.99
Model Railway Points Controller receiver	09101242	£2.99
Mains Power-Up Sequencer	10108231	£11.99
JANUARY 2025		
USB to PS/2 Converter for VGA PicoMite	07111231	£3.99
ps2x2pico USB to PS/2 Converter	07111232	£3.99
Compact Microphone Preamplifier (case version)	01110231	£5.99
Compact Microphone Preamplifier (integrated version)	01110232	£5.99
DECEMBER 2024		
Raspberry Pi Clock Radio PCB pair	19101241/2	£12.99
VGA PicoMite (blue)	07107221	£5.99
Secure Remote Switch receiver	10109231	£4.99
Secure Remote Switch transmitter (module version)	10109232	£2.99
Secure Remote Switch transmitter (discrete version)	10109233	£2.99
NOVEMBER 2024		
Ideal Diode Bridge Rectifiers (28mm square)	18101241	£3.99
Ideal Diode Bridge Rectifiers (22mm square)	18101242	£3.99
Ideal Diode Bridge Rectifiers (4-pin SIL)	18101243	£3.99
Ideal Diode Bridge Rectifiers (tiny SOT-23)	18101244	£3.99
Ideal Diode Bridge Rectifiers (standalone SMD)	18101245	£4.99
Ideal Diode Bridge Rectifiers (standalone TH; 2oz copper)	18101246	£6.99
Multi-channel Volume Control (volume PCB)	01111221	£4.99
Multi-channel Volume Control (control PCB)	01111222	£4.99
Multi-channel Volume Control (OLED PCB, black)	01111223	£4.99
OCTOBER 2024		
TQFP-32 Programming Adaptor	24108231	£4.99
TQFP-44 Programming Adaptor	24108232	£4.99
TQFP-48 Programming Adaptor	24108233	£4.99
TQFP-64 Programming Adaptor	24108234	£4.99
2m VHF FM Test Signal Generator	06107231	£4.99
SEPTEMBER 2024		
PIC Programming Adaptor	24105231	£4.99
Salad Bowl Speaker crossover	01109231	£7.99
AUGUST 2024		
Arduino ESR Meter combined L/C-ESR	04106182	£9.99
Arduino ESR Meter separate ESR	04106183	£7.99
Watering System Controller	15110231	£11.99

PROJECT	CODE	PRICE
JULY 2024		
Dynamic NFC Tag (22 × 31mm)	06101231	£5.95
Dynamic NFC Tag (86 × 54mm)	06101232	£7.95
Reciprocal Frequency Counter	CSE230101C	£8.95
Reciprocal Frequency Counter front panel (black mask)	CSE230102	£7.95
Pi Pico-based Thermal Camera	04105231	£6.95
JUNE 2024		
Loudspeaker Test Jig	04106231	£12.95
MAY 2024		
GPS-Disciplined Oscillator	04103231	£12.95
Dual RF Amplifier	CSE220602A	£5.95
Songbird	08103231	£7.95
APRIL 2024		
Basic RF Signal Generator	CSE221001	£8.99
Basic RF Signal Generator front panel (black solder mask)	CSE220902B	£9.99
Silicon Chirp Cricket	08101231	£7.99
Test Bench ‘Swiss Army Knife’	04110221	£9.99
MARCH 2024		
Digital Volume Control Potentiometer SMD version	01101231	£8.95
Digital Volume Control Potentiometer through-hole version	01101232	£9.95
Advanced SMD Test Tweezers body	04106221	£6.95
Advanced SMD Test Tweezers legs (pair)	04106212	£7.95
FEBRUARY 2024		
Active Mains Soft Starter	10110221	£10.95
JANUARY 2024		
Q Meter	CSE220701	£8.95
Q Meter front panel (black solder mask)	CSE220704	£7.95
Raspberry Pi Pico W BackPack	07101221	£8.95
DECEMBER 2023		
Digital Boost Regulator	24110224	£8.95
Dual-Channel Power Supply for Breadboards	04112221	£11.95
Display Adaptor for the Breadboard PSU	04112222	£9.95
NOVEMBER 2023		
LC Meter Mk3	CSE220503C	£9.95
DC Supply Filter for vehicles	08108221	£8.95
Discrete Audio Op Amp PCB	AO1-JUL23	£9.95
Discrete Audio Op Amp PCB + essential components	AO2-JUL23	£17.95
OCTOBER 2023		
Buck/Boost Charger Adaptor	14108221	£9.95
PIC Breakout Board for SOIC parts	24110225	£5.95
PIC Breakout Board for DIP parts	24110222	£5.95
AVR64DD32 Breakout board	24110223	£5.95
Automatic Train Controller	09109221	£6.95
Chuff Sound module	09109222	£6.95
SEPTEMBER 2023		
Mini LED Driver	16106221	£7.95
New GPS-Synchronised Clock	19109221	£9.95
AUGUST 2023		
110dB RF Attenuator	CSE211003	£9.95
Universal Battery Charge Controller (2023 update)	14107192	£9.95
Wide-Range OhmMeter	04109221	£12.95
JULY 2023		
Multimeter Checker/Calibrator	04107221	£10.95

PCBs for most recent *PE/EPE* constructional projects are available. From the July 2013 issue onwards, PCBs with eight-digit codes have silk screen overlays and, where applicable, are double-sided, have plated-through holes, and solder mask. They are similar to photos in the project articles. Earlier PCBs are likely to be more basic and may not include silk screen overlay, be single-sided, lack plated-through holes and solder mask.

Always check price and availability in the latest issue or online. A large number of older boards are listed for ordering on our website.

In most cases we do not supply kits or components for our projects. For older projects it is important to check the availability of all components before purchasing PCBs.

Magazine back issues are available – see https://www.electronpublishing.com/product-category/pe-back-issues-print-and-digital/

Double-sided | plated-through holes | solder mask

PROJECT	CODE	PRICE
JUNE 2023		
Arduino Programmable Load	04105221	£8.95
Buck-Boost LED Driver	16103221	£8.95
MAY 2023		
Precision AM-FM DDS Signal Generator	CSE211002	£14.95
Improved SMD Test Tweezers programmed PIC	0410621PIC	£12.95
AO 2x-dual-to-quad through-hole (pack of five)	AO1-MAY23	£6.95
AO 2x-dual-to-quad SMD (pack of five)	AO2-MAY23	£6.95
AO 2x-single-to-dual through-hole (pack of five)	AO3-MAY23	£6.95
AO 2x-single-to-dual SMD (pack of five)	AO4-MAY23	£6.95
APRIL 2023		
500W Amplifier Module	see p.22, April 2023	
Clipping Indicator (per channel)	01112211	£7.95
CD Welder Power Supply (one needed)	29103221	£9.95
CD Welder Controller (one needed)	29103222	£9.95
CD Welder Energy Storage module (several needed)	29103223	£7.95
AO Universal Dual Op Amp Board	AO1-APR23	£9.95
AO Stereo RIAA precision passives kit for dual op amp	AO2-APR23	£8.95
MARCH 2023		
Pico BackPack	07101221	£9.95
Semaphore Signal (controller)	09103221	£7.95
Semaphore Signal (blade)	09103222	£5.95
FEBRUARY 2023		
Tesla Coil driver board	26102221	£9.95
Tesla Coil potentiometer board	26102222	£5.95
Cooling Fan Controller & Loudspeaker Protector	01102221	£8.95
Remote Gate Controller	11009121	£12.95
JANUARY 2023		
Classic LED Metronome – 8-LED	23111211	£7.95
Classic LED Metronome – 10-LED	23111212	£8.95
Multi-Channel Speaker Protector – 6 channel	01101221	£9.95
Multi-Channel Speaker Protector – 4 channel	01101222	£7.95
Remote Control Range Extender – IR-to-UHF	15109212	£5.95
Remote Control Range Extender – UHF-to-IR	15109211	£7.95
AO Universal Single Op Amp Board	AO1-JAN23	£7.95
DECEMBER 2022		
Hummingbird Amplifier	01111211	£9.95
SMD Trainer PCB	29106211A	£8.95
SMD Trainer PCB + parts	29106211B	£13.95
NOVEMBER 2022		
USB Cable tester – main PCB	04108211	£12.95
USB Cable tester – front panel	04108212	£5.95
USB Cable tester – optional panel	SC5970	£5.95
Model Railway Carriage Lights – PCB	09109211	£6.95
AO trnsfmr PCB – standard VTX-A range	VTX-101-007	£6.95
AO trnsfmr PCB – dual-outline VTX102-3001/101-3002	VTX-Dual	£6.95
OCTOBER 2022		
SMD Test Tweezers – PCB and pair of tweezer arms	04106211-2	£11.95
SMD Test Tweezers – programmed PIC12F1572-I/SN	0410621A	£7.95
Tele-com	12110211	£12.95
SEPTEMBER 2022		
Touchscreen Digital Preamp – main board	01103191	£12.95
Touchscreen Digital Preamp – adaptor board pair	01103192	£5.95
20A DC Motor Speed Controller	11006211	£9.95

PROJECT	CODE	PRICE
AUGUST 2022		
Multi-purpose Battery Manager – I/O Expander	11104212	£5.95
Multi-purpose Battery Manager – Switch Module	11104211	£8.95
Simple MIDI Music Keyboard (for 8 switches)	23101213	£6.95
Nano Pong	08105212	£7.95
JULY 2022		
Silicon Labs AM/FM/SW Radio	CSE210301C	£10.95
Level Crossing Controller	09108211	£6.95
JUNE 2022		
Full-wave Motor Speed Controller	1010221	£8.95
PIC Programming Helper for 8-pin PICs only	24106211	£7.95
PIC Programming Helper for 8, 14 or 20-pin PICs	24106212	£10.95
Advanced GPS Computer	05102211	£9.95
MAY 2022		
Bus board PCB for Analogue Vocoder	AO1-MAY22	£10.95
Complete set of 14 PCBs for Analogue Vocoder	AO2-MAY22	£97.95
Programmed EEPROM for Digital FX Unit	FX1-MAY22	£10.95
Programmed PIC for Digital FX Unit w/ potentiometer	FX2-MAY22	£8.95

For the many earlier PCBs that we stock please see the PE website: www.electronpublishing.com

PE/EPE PCB SERVICE

Order Code	Project	Quantity	Price

Name ....

Address ....

....

Tel ....

Email ....

Card No ....

Valid From ....Expiry Date ....

Card Security No ....

You can also order PCBs by phone, email or via the shop on our website: www.electronpublishing.com

No need to cut your issue – a copy of this form is just as good!

All prices include VAT and UK p&p. Add £4 per *project* for post to Europe; £5 per *project* outside Europe.

Orders and payment should be sent to:

Practical Electronics, Electron Publishing Ltd
113 Lynwood Drive, Merley, Wimborne, Dorset BH21 1UU
Tel 07918 614662   Email: pe@pemag.au
Online Shop: www.electronpublishing.com

Sorry, cheques cannot be accepted as we are based in Australia and our UK bank account does not allow cheque deposits.

NOTE: Most boards are in stock and sent within seven days of order receipt. Please allow up to 28 days delivery if we need to restock.

Practical Electronics PCB SERVICE

PROJECT	CODE	PRICE
APRIL 2022		
64-key MIDI Matrix shield	23101211	£8.95
64-key MIDI Matrix switch board	23101212	£11.95
High-current Battery Balancer	14102211	£10.95
Digital FX Unit – using potentiometer	01102211	£9.95
Digital FX Unit – using BCD switch	01102212	£9.95
Universal Audio PSU	AO1-APR22	£11.95
MARCH 2022		
Mini Isolated Serial Link	24102211	£5.95
Busy Loo Indicator	16112201	£5.95
Analogue Vocoder – Band-pass filter board	AO1-MAR22	£9.95
Analogue Vocoder – HP/LP filter board	AO2-MAR22	£9.95
FEBRUARY 2022		
Arduino-based Power Supply	18106201	£9.95
Battery Monitor Logger	11106201	£10.95
Electronic Wind Chimes	23011201	£10.95
Analogue Vocoder – Driver Amplifier	AO-FEB22	£8.95
JANUARY 2022		
Vintage battery Radio Li-ion Power Supply	11111201	£9.95
MiniHeart: A Miniature Heartbeat Simulator	01109201	£8.95
DECEMBER 2021		
AM/FM/SW Digital Receiver	CSE200902A	£13.95
Balanced Input and Attenuator for USB CODEC	01106202	£11.95
NOVEMBER 2021		
Dual Battery Lifesaver	11111202	£6.95
OCTOBER 2021		
Mini Wi-Fi LCD BackPack	24106201	£8.95
SEPTEMBER 2021		
USB SuperCodec PCB	01106201	£14.95
Audio DDS Oscillator PCB	01110201	£5.95
Audio DDS Oscillator rotary encoder	01110201-ENC	6.95
Programming Adaptor Board for Audio DDS Oscillator	01110202	£5.95
High-power Ultrasonic Cleaner main PCB	04105201	£14.95
High-power Ultrasonic Cleaner front-panel PCB	04105202	
Night Keeper Lighthouse PCB	08110201	£5.95
AUGUST 2021		
Ol' Timer PCB	19104201	£11.95
Ol' Timer 8x8 RGB LED module using WS2812B	19104201-88	£8.95
Ol' Timer set of acrylic case pieces and spacer	19104201-ACR	£8.75
Ol' Timer DS3231 RTC IC wide SOIC-16	19104201-RTC	£5.95
Wideband Digital RF Power Meter	04106201	£9.75
Switchmode 78xx regulators ***(PACK of 5!)***	18105201	£7.95
Cool Beans SMAD display	CB-AUG21	£11.95
JULY 2021		
ATtiny816 Breakout / Dev Board w/ Capacitive Touch	24110181	£9.75
IR Remote Control Assistant (Jaycar version)	15005201	£8.95
IR Remote Control Assistant (Altronics version)	15005202	£8.95
PIC18F Development Board	PNM-JUL21	£12.95
Microphone Preamplifier	AO-JUL21	£11.95
JUNE 2021		
Roadies' Test Signal Generator (surface-mount ver.)	01005201	£8.95
Roadies' Test Signal Generator (through-hole version)	01005202	£9.95
Touchscreen Wide-range RCL Box (Resistor module)	04104201	£18.95
Touchscreen Wide-range RCL Box (Ind/Cap module)	04104202	
KickStart Part 3 – Gyrator-based Audio Filter	KS3-2021	£7.95

PROJECT	CODE	PRICE
MAY 2021		
7-Band Equaliser (Mono)	01104201	£8.95
7-Band Equaliser (Stereo)	01104202	£10.95
Car Altimeter	05105201	£7.95
APRIL 2021		
Reflow Oven – DSP Active Crossover (CPU)	01106193	£19.95
Reflow Oven – DSP Active Crossover (Front panel)	01106195	
Reflow Oven – DSP Active Crossover (LCD)	01106196	
Frequency Reference Signal Distributor	CSE200103	£8.95
MARCH 2021		
Nutube Guitar Effects Pedal	01102201	£12.95
Programmable Thermal Regulator (Peltier Interface)	21109181	£18.95
Programmable Thermal Regulator (Peltier Driver)	21109182	
Tunable HF Preamp	CSE190502	£8.95
FEBRUARY 2021		
4G Remote Monitoring	27111191	£9.95
JANUARY 2021		
Nutube Valve Preamplifier	01112191	£12.95
Arduino DCC Controller	09207181	£10.95
DECEMBER 2020		
Pseudo-Random Sequence Generator	16106191	£7.95
Clever Charger	14107191	£11.95
PE Theremin Amplifier	AO-1220-01	£8.95
NOVEMBER 2020		
LED Christmas Tree (1 off)	16107181-1	£6.95
LED Christmas Tree (4 off)	16107181-2	£14.95
LED Christmas Tree (12 off)	16107181-3	£24.95
LED Christmas Tree (20 off)	16107181-4	£34.95
USB/SPI Interface Board	16107182	£5.95
45V/8A Power Supply PCB *plus* acrylic spacer	18111181	£14.95
45V/8A Power Supply five-way display bezel	18111181-BZ	£3.95
Five-way LCD Panel Meter/Display	18111182	£7.95
OCTOBER 2020		
Digital Audio Millivoltmeter	04108191	£9.95
Precision Signal Amplifier	04107191	£7.95
SEPTEMBER 2020		
PE Theremin PSU	AO-0920-01	£5.95
PE Theremin PSU transformer	AO-0920-02	£7.95
Micromite Explore-28	07108191	£6.95
Ultrabrite LED Driver	16109191	£6.95
AUGUST 2020		
Micromite LCD BackPack V3	07106191	£9.95
Steering Wheel Audio Button to Infrared Adaptor	05105191	£7.95
JULY 2020		
AM/FM/CW Scanning HF/VHF RF Signal Generator	04106191	£13.95
Speech Synthesiser with the Raspberry Pi Zero	01106191	£5.95
PE Mini-organ PCB	AO-0720-01	£14.95
PE Mini-organ selected parts	AO-0720-02	£8.95
High-current 12V Battery Isolator – control	05106191	£6.95
High-current 12V Battery Isolator FET (2oz)	05106192	£9.95
JUNE 2020		
Arduino breakout board – 3.5-inch LCD Display	24111181	£6.95
Six-input Audio Selector main board	01110191	£10.95
Six-input Audio Selector switch panel board	01110192	

Silicon Chip kits & parts for projects in this issue (posted from Australia):

USB-C Serial Adaptor kit	https://siliconchip.au/Shop/20/6652	AUD $24 (about £12) + P&P
DC Supply Protector kit (adjustable SMD version)	https://siliconchip.au/Shop/20/6948	AUD $21 (about £10.50) + P&P
DC Supply Protector kit (adjustable TH version)	https://siliconchip.au/Shop/20/6949	AUD $27 (about £13.50) + P&P
DC Supply Protector kit (fixed TH version)	https://siliconchip.au/Shop/20/6950	AUD $24 (about £12) + P&P

20% VAT is included in the above prices (it will be removed for customers outside the UK). Postage & packing cost is typically around £12-15. Please allow 1-2 weeks for delivery (longer in some cases).

CLASSIFIED ADVERTISING

Practical Electronics

If you want your advertisements to be seen by the largest readership at the most economical price then our classified page offers excellent value.

The rate for semi-display space is £10 per centimetre high, with a minimum height of 2·5cm. All semi-display advertisements have a width of 5.5cm. The prepaid rate for classified adverts is 40p per word (minimum 12 words).

For rates, further information on display and classified advertising or to book an ad, please email us at pe@pemag.au

Practical Electronics reaches more UK readers than any other UK monthly hobby electronics magazine. Our sales figures prove it.

We have been the leading monthly magazine in this market for more than thirty years.

BOWOOD ELECTRONICS LTD
Suppliers of Electronic Components
www.bowood-electronics.co.uk
Unit 10, Boythorpe Business Park, Dock Walk, Chesterfield, Derbyshire S40 2QR. Sales: 01246 200 222
Send large letter stamp for Catalogue

MISCELLANEOUS

VALVES AND ALLIED COMPONENTS? For free stock list and/or advice, please contact me: geoffdavies337@gmail.com Telephone: 01788 574774

PIC DEVELOPMENT KITS, DTMF kits and modules, CTCSS Encoder and Decoder/Display kits. Visit www.cstech.co.uk

Electrical Industries Charity (EIC)
We help people working in the electrical, electronics and energy community as well as their family members and retirees.
We use workplace programmes that give the industry access to financial grants and a comprehensive range of free and confidential services.
www.electricalcharity.org
electrical industries charity

COMPONENTS AS JOB LOT Large stock. Transistors, Diodes, Logic, Memory and Analogue parts. 1000s of passive items. Catalogued and sorted. 'New Old Stock' in racks. Email: rojobarn@gmail.com

COAST ELECTRONICS
BREAKOUTS-COMPONENTS-CONTRACT DESIGN-3D PRINTER PARTS-MUSICAL-MICROCONTROLLERS
WWW.COASTELECTRONICS.CO.UK

Andrew Kenny – Qualified Patent Agent
EPO UKIPO USPTO
Circuits Electric Machinery Mechatronics
Web: www.akennypatentm.com
Email: Enquiries@akennypatentm.com
Tel: 0789 606 9725

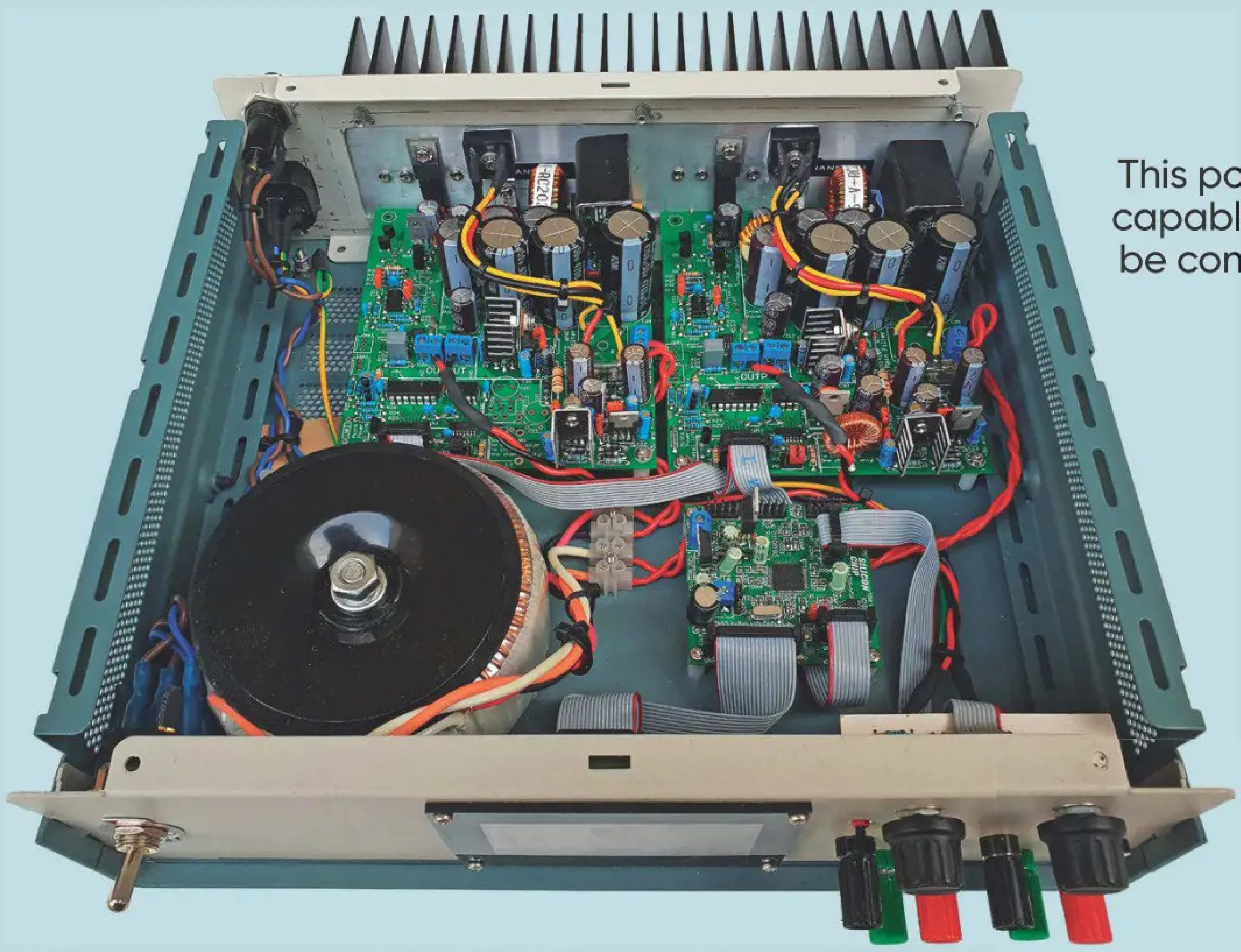
ADVERTISING INDEX

BACK ISSUE PDFs	63
BASIC MICRO	24
ENIGMA COMPONENTS SHOP	17
HAMMOND ELECTRONICS Ltd.	17
MICROCHIP TECHNOLOGY	Outer back cover
PCBWAY	Inner front cover
POLABS	45
STEWART OF READING	34
WIRELESS FOR THE WARRIOR BOOKS	24

Advertising mailing address
Electron Publishing Australia
PO Box 194
Matraville NSW 2036
Australia
Email pe@pemag.au
Web www.electronpublishing.com

For editorial contact details, see page 2.

Next Month – in the June issue



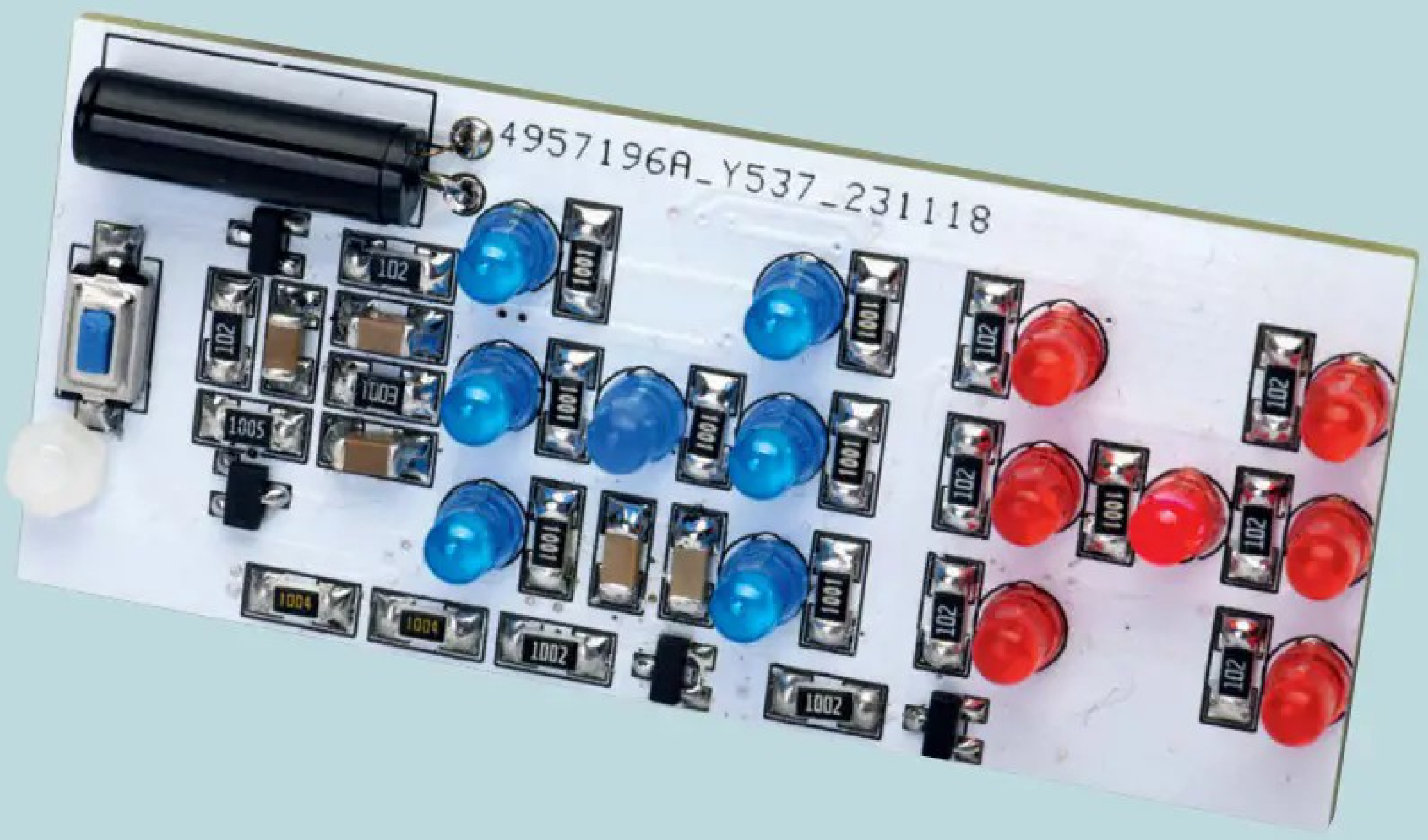
Dual Hybrid Power Supply

This power supply has two separate outputs, each capable of delivering up to 25V DC at 5A. They can be connected in series and ganged to form a dual tracking supply.

The Styloclone

An easy to build (and play) musical instrument!

Silicon Chip
Styloclone



Dual Mini LED Dice

It 'rolls' two six-sided dice at once using just a few logic ICs and can be built with SMD or through-hole LEDs.

On sale 1st of May, 2025

Content may be subject to change

Practical Electronics

Subscriptions hotline

To subscribe or renew online, paying by credit/debit card or PayPal, please visit our shop at: www.electronpublishing.com

For those wishing to pay by direct debit, cheque or over the telephone, please call 'Select Publisher Services' on 01202 087631, or send a cheque (payable to: 'Select Publisher Services Ltd') with your details to:

Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Remember, we print the date of the last issue of your current subscription in a box on the address sheet that comes with your copy.

Digital subscribers, please call 07918 614662 or visit: www.electronpublishing.com

Published on approximately the first Thursday of each month by Silicon Chip Publications, U1 234 Harbord Road Brookvale NSW 2100 Australia. Printed in England by The Manson Group, Reynolds House, 8 Porters Wood, Valley Road Industrial Estate, St Albans AL3 6PZ. Distributed by Seymour, 86 Newman St, London W1T 3EX. Subscriptions UK: £33.99 (6mo); £59.99 (12mo.); £114.99 (2yrs). EUROPE: airmail service, £38.99 (6mo); £69.99 (12mo); £129.99 (2yrs). REST OF THE WORLD: airmail service, £46.99 (6mo); £84.99 (12mo); £164.99 (2yrs). Make payments to 'Practical Electronics', Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH United Kingdom. Email: pesubs@selectps.com. PRACTICAL ELECTRONICS is sold subject to the conditions that it shall not, without the written consent of the Publishers first having been given, be lent, resold, hired out or otherwise disposed of by way of Trade at more than the recommended selling price on the cover, and that it shall not be lent, resold, hired out or otherwise disposed of in a mutilated condition or in any unauthorised cover by way of Trade or affixed to or as part of any publication or advertising, literary or pictorial matter.

Did you know our online shop now sells the current issue of *PE* for £5.99 inc. p&p?

Practical Electronics
The UK's premier electronics, computing and maker magazine
Circuit Surgery
Digital high-pass, band-pass & band-stop filters
Audio Out
An amplifier that switches between Class-B and Class-A
Precision Electronics
Quantifying and dealing with noise
Protecting devices from damaging voltages
DIY USB-C serial adaptor
Skill Tester 9000
A new take on an old game
PLUS!
Techno Talk – the latest electronics technology
Cool Beans – creating & optimising display algorithms
Sinclair Calculators – 1970s innovation in the UK
Adding solar charging to a Renault Kangoo electric van
May 2025 £5.99
www.electronpublishing.com
@practicalelec
practicalelectronics
Precision Electronics
Analog signal switching techniques and tips
Calibrating MEMS Microphones
Apr 2025 £5.99

You read that right! We now sell the current issue of your favourite electronics magazine for exactly the same price as in the High Street, but we deliver it straight to your door – and for UK addresses we pay the postage. No need to journey into town to queue outside the newsagent. Just go to our website, set up an account in 30 seconds, order your magazine and we'll do the rest.

www.electronpublishing.com

Code Smarter, Faster and Better—Meet MPLAB® AI Coding Assistant

Discover Our New, Free AI Coding Assistant Powered by Continue

The MPLAB AI Coding Assistant is here to revolutionize your coding experience. The MPLAB AI Coding Assistant is a customized version of the Continue extension from Microsoft® Visual Studio® Code (VS Code®) Marketplace, one of the leading open-source AI code assistants on the market today. This powerful tool is tailored specifically for users developing Microchip products and facilitates the development of Microchip-specific features, as well as new slash commands and shortcuts or common use cases.

Explore some of the key features that enhance your development process:

- **Pre-Configured Chatbot:** Interact with a Microchip-trained chatbot and ask Microchip-specific questions.
- **Autocomplete:** Get online code suggestions as you type to speed up development and minimize errors.
- **Search:** Quickly access Microchip datasheets directly without leaving the editor.
- **Code Generation and Review:** Generate, review, explain and comment on code effortlessly.
- **Integrated Development:** Develop and edit code without leaving the editor, with the chatbot conveniently located in the sidebar.

Users can do all this without leaving the editor, as the chatbot—once installed—sits in the sidebar. Enhance your coding experience today with our new MPLAB AI Coding Assistant, which is continuously updated with Microchip-specific information and take your development to the next level with seamless integration, real-time support and increased productivity.

MPLAB® EXTENSIONS

MICROCHIP

www.microchip.com/MPLABCodeAI

The Microchip name and logo and the Microchip logo are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. All other trademarks are the property of their registered owners.
© 2025 Microchip Technology Inc. All rights reserved.